The Fifth International Conference on Business Process Management 2007

Enabling Change and Innovation

WORKSHOP ON

BPM IN HEALTHCARE

INFORMAL PROCEEDINGS

Hosted by

Queensland University of Technology
Brisbane Australia

Formal proceedings to be published by Springer-Verlag

Lecture Notes in Computer Science
LNCS LNAI LNBI

Contents

Contents

Careflow: Theory and Practice

John Fox[1,3] and Robert Dunlop[2]

[1]Department of Engineering Science, University of Oxford, Oxford, UK
[2]*Infer*Med Ltd, London, UK
[3]UCL Medical School and Royal Free Hospital, London, UK

Abstract. In this presentation we shall review different ways of describing the *processes* of delivering patient care, and relate these to traditional workflow in business processes, and the concept of "careflow" proposed by Panzarasa and her colleagues in Pavia. We shall discuss the problem of designing a careflow application as a form of *process modelling*, to be contrasted with older paradigms ranging from rule-based alerts and reminders to Petri nets and Critical path analysis. We shall also consider the need for specialised *formalisms* for describing clinical processes, drawing on experience with workflow languages (e.g. BPEL4WS, BPMN), guideline modelling languages (e.g. GLIF, PRO*forma*), AI planning languages (e.g. PDDL, OCL) and "agent" programming systems (e.g. LALO, 3APL). The adoption of clinical workflow technology will greatly benefit from the availability of appropriate languages for declaratively representing processes of care. To explore some of the requirements for a clinical workflow technology we will review and critique the PRO*forma* process modelling language and the Arezzo® and Tallis toolsets which use it. The discussion will be illustrated with deployed applications and operational prototypes.

Guideline Models, Process Specification, and Workflow

Samson W. Tu, MS

Stanford University School of Medicine
Stanford, CA 94305-5479, USA

Abstract. Many clinical practice guidelines use flowcharts to aid the description of recommendations specified in the guidelines. Similarly a number of computer-interpretable guideline formalisms use graphical task networks to organize knowledge formalized in these models. However, the precise meaning encoded in these graphical structures is often unclear. In this presentation, I will survey some of the computer-interpretable formalisms and analyzed the graphical representations that are used to express process information embodied in clinical guidelines and protocols. I will argue that we can distinguish a number of process types: (1) flowcharts for capturing problem-solving processes, (2) disease-state maps that link decision points in managing patient problems over time, (3) plans that specify sequences of activities that contribute toward a goal, (4) workflow specifications that model care processes in an organization, and (5) computational processes that generates recommendations for specific clinical situations. These process types may be related to each other. A plan of actions, for example, may be mapped to a workflow process when its actions are assigned to healthcare providers playing different roles. A flowchart may depict decisions and actions that are performed over time. Furthermore, a guideline formalism may not make a commitment to the nature of processes being modeled. Its process-specification language may be used to encode different types of processes. Nevertheless, understanding the nature of process being modeled is crucial when it comes to enacting the encoded guidelines and protocols to provide decision support in clinical workflow. A process description that models the problem-solving steps depicted in a narrative guideline, for example, may contain steps that are not appropriate as part of human-computer interactions in a busy clinic.

Restrictions in Process Design: A Case Study on Workflows in Healthcare

Jörg Becker, Christian Janiesch

European Research Center for Information Systems,
Leonardo Campus 3, 48149 Münster, Germany
{becker, janiesch}@ercis.de

Abstract. Automating existing processes is as paving cow path compared to major business process reengineering. However, this rather radical approach is not suitable for all business fields. It requires the freedom to modify organizational structures and free core business processes from non-value adding activities. In sectors like healthcare, there are a variety of legal restrictions and treatment guidelines practitioners have to comply with. Hence, freedom to reorganize the organization and to omit non-value adding activities is heavily compromised. In this paper we present findings from a case study that exemplify restrictions in process reorganization and suggest utilizing more moderate approaches to process management.

Keywords: Process design, design restrictions, process management, workflow management, healthcare, infection control.

1 Introduction

Hammer and Champy [8] see the practice of automating existing processes as "paving cow path" compared to major Business Process Reengineering (BPR). While it is desirable to take the blinkers off to free oneself from restraints of everyday procedures this rather radical approach is not suitable for all business fields. BPR requires the freedom to modify organizational structures and free core business processes from non-value adding activities. This requires introducing radical changes as well as new procedures. In business sectors like healthcare, there are a variety of legal restrictions and treatment guidelines practitioners have to comply with [21, 22]. Hence, freedom to reorganize the organization and to omit non-value adding activities as well as to change mandatory procedures and existing medical information systems (IS) is heavily compromised.

Thus, in healthcare one needs to utilize the less radical principles of Business Process Management (BPM) [1, 25]. Short and precise projects and continuous improvement offer a passable way despite a restrictive environment and legal pre-requisites. Through small iterations potential for process optimization, i.e. reduce cost, free staff from routine work, and improve patient safety without reengineering the company can be achieved. This paper presents a case, performed to show how BPM and commercial off-the-shelf workflow software contribute to lower the

frequency of human errors in healthcare [10, 11] by introducing gradual change. The goal of the case study was to improve efficiency of an existing controlling process for hospital acquired infections (HAI).

The structure of the paper is as follows: First, a short literature review summarizes relevant facts on BPR and BPM as well as workflow management. In Section 3 an introduction to healthcare and clinical processes is given as characterization of the project. Section 4 comprises the case study including details on restrictions in process design as well as on the subsequent workflow implementation. The paper closes with conclusions to an outlook.

2 Fundamentals of Processes and Workflows

Processes are generally seen as any activity performed within a company or an organization [14]. In the context of this work, we define a process as "a completely closed, timely and logical sequence of activities which are required to work on a process-oriented business object" [1]. Consequently, a business process is considered as a special kind of process that is directed by business objectives of a company and by the business environment [1]. Business processes can be further classified into value creating core business processes and not value adding supplementary processes. Whereas core business processes are considered to contain corporate expertise and produce products or services that are delivered to customers [9, 16], supplementary business processes facilitate the ongoing operation of the core processes. This distinction is not intended to be always selective as one business process might be a core business process for one product and a supplementary business process for another [1].

The practice of business process reengineering, which emerged in the early 1990s, is seen as fundamental rethinking and radical redesign of business processes. In doing so dramatic improvements in critical, contemporary measures of performance, such as cost, quality, service, and speed can be achieved [8]. This kind of greenfield project, however, does not consider any existing operational sequences or organizational structures during the building of new processes at all. Furthermore, BPR targets the overall process perspective in one single shot rather than iteratively and continuously optimizing process performance. BPM on the other hand serves the planning, controlling, and monitoring of intra- and inter-organizational processes with regards to existent operational sequences and structures, in a consistent, continuous iterative way of process improvement [1].

In dependence to processes, workflows can be seen as part of a work process that contains the sequence of functions and information about the data and resources involved in the execution of these functions [2]. Workflows are an automated representation of a whole or part of a business process. Procedural rules define documents, information or tasks, which are to be passed from one participant to another for action [27].

To-be process models are used as sources to implement workflows. Therefore, process models need to be transformed into workflow models. Process models, however, primarily serve organizational (re-)design whereas workflow models focus

on implementing IT support. That is why process models integrate functions in a lower level of granularity than workflow models [2].

While creating workflow models, workflow relevant data is required for the refinement of functions. In consequence, the necessity for a detailed specification of data needed during the execution of activities and data needed to create mathematic routing conditions emerges. Also, criteria when to initiate and when to terminate a workflow and how to handle errors are to be defined [28].

Workflow Management (WfM) aims at providing this automated process execution where the transitions between the individual activities are controlled by a Workflow Management System (WfMS) [28]. If an activity cannot be automated, a WfMS is concerned with demanding input from users while providing all necessary information needed to make a decision.

3 Healthcare and Clinical Processes

Healthcare providers are under constant pressure to reduce costs while the quality of care is to be improved [10]. Expenses for patient treatment and pharmaceuticals are relentlessly rising whereas reimbursements, refunded by insurance providers, are coupled to diagnosis-related groups and fixed [7].

Clinical processes can be classified as generic process patterns or medical treatment processes [12, 15]. Both types of processes may be designed and executed cross-department as well as cross-company. Generic process patterns help to coordinate healthcare processes among different people and organizational units. Medical treatment processes are the representation of an actual care process, which are considered to be the core processes of healthcare facilities. These processes highly depend on medical knowledge and case specific decisions [15]. Clinical process decisions are made by interpreting patient specific data according to clinical knowledge. In order to provide clinical decision support, patient specific data needs to be consolidated and a recallable representation of clinical knowledge needs to be provided in medical IS. The cooperation of clinical knowledge and complex decision support allows the implementation of treatment guidelines in highly flexible processes. Flexibility is required since treatment of patients is likely to differ from patient to patient. In consequence, medical treatment processes need to be quickly adaptable [12]. Medical treatment processes can be further described as a diagnostic-therapeutic cycle. Main components of the diagnostic-therapeutic cycle are: observation, reasoning, and action. These stages are iterated until no further action needs to be taken, i.e. the patient no longer requires treatment [12].

The historical evolvement of heterogeneous IS in healthcare may be due to a lack of expertise in implementing systems, missing investment abilities, but also the development of technology needs to be taken into account [13].

Infection Control (IC) is the process of preventing hospital acquired infections (HAI) by isolating sources of infections and limiting their spread. Nowadays, HAI are by far the most common complications affecting hospitalized patients or intensive care patients [3, 5]. Approximately 2 million patients are affected each year and costs add up to estimated $4.5 to $5.7 billion per year [4]. Identification of HAI typically

involves testing of specimen in a laboratory. In addition, nurses working at nurse stations need to get specimen, physician need to order the specimen tests, and finally Infection Control Practitioners (ICP) need to ensure that all precautions have been taken, if a specimen was tested positive.

4 A Case Study on Workflows in Healthcare

4.1 Case Study Scenario

The case was performed in a major healthcare facility in the U.S. The facility consists of multiple independent hospitals. More than 7500 employees are employed at four sites, medical staff counts around 1000 physicians throughout the organization. Overall, almost 1000 beds are available for inpatient care. The scope of the project was to analyze the current IC process, suggest possible improvements through workflow, and finally enhance the current IT solution to increase process efficiency. The uniqueness of this project was rooted in the application service providing (ASP) environment [6, 19].

The team for this subproject consisted of five method experts for process modeling and implementation and six domain experts at the healthcare facility for analysis, test, and evaluation. Staff for technical support (ASP, rule engine) was provided by the overall project management. Process modeling, implementation and pre- and post-metrics took six month; build, test, and integration of the workflow needed to be done in only twelve weeks.

Analog to the theoretically exploration in the previous sections, the actual freedom to restructure processes or the organization was found to heavily compromised by legal restrictions and health care guidelines. Although this became apparent already during the first stage of analysis, consensus was achieved to pursue the project even though potential for optimization could not be fully utilized. It was agreed that a workflow focused pilot project would provide essential knowledge for more complex projects to come.

The software architecture is best described as a three tier, client server architecture built according to principles of service oriented architecture. The architecture consists of the web application tier, the top layer, constituted by web application servers running a user interface. The application tier, the middle layer, is constituted by application servers, a rules engine, and the WfMS.

We used Soarian® as medical IS in this project [18]. The Soarian® environment uses the third part WfMS TIBCO® Staffware Process Suite [23]. The WfMS can use services provided by the medical IS to add and remove items to/from user specific worklists. Users of the medial IS can trigger events, hence invoke the workflow engine to perform actions on demand. Whereas the WfMS evaluates simple routing conditions by itself, complex clinical conditions need to be evaluated with respects to clinical knowledge and patient specific data. Therefore, a rules engine based on Arden Syntax [17] can be used by the WfMS. This rules engine evaluates complex decision in clinical workflows.

4.2 As-is Analysis

The IC process at the customer consists of two separate process fragments, synchronized over paper reports. The first part of the process starts as soon as a specimen is tested, if the patient to which the specimen is assigned to, is an inpatient at the facility. First, the lab result needs to be checked whether it indicates a positive statement of a HAI. This is done by the laboratory, which has already performed the test of the specimen. Once a lab result indicating a positive infection statement has been identified, an employee working at the laboratory calls the floor the patient is located on. In doing so, the nurse station is notified about an infectious patient and responsibility for putting the patient in isolation is passed to the nurse station. After the nurse station has initiated necessary tasks to isolate the patient, the first part of the IC process ends, as soon as the laboratory has completed the documentation of the lab result in the lab system. The process depicted in Figure 1 illustrates a structured overview of this part of the IC process.

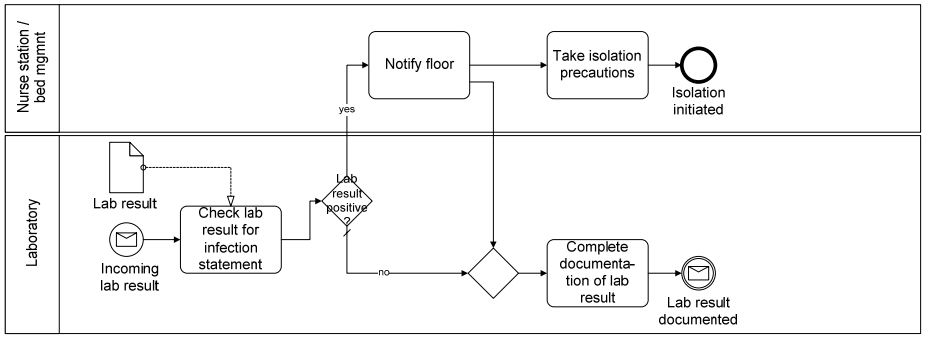

Fig. 1. As-is notification process.

The second part of the IC process is a rather a controlling process. U.S. hospitals are required to have ICP supervising the handling of infections at each facility. Furthermore, each hospital must report the amount of infection occurrences on a yearly basis [26]. As for this facility, specific reports for each infectious disease were created on a monthly or even daily basis. These reports are the starting point of the second fragment of the IC process (cf. Figure 2). Once a report of an infectious disease is received by an ICP, it needs to be checked for infection statements. This task results in a list of patients that need a follow-up ensuring that patients who require isolation are actually put in isolation. During daily tours, the ICP does not only check if infection precautions have been taken for infectious patients but also controls, if the infection statement is transferred to the patient's chart. If a patient is not put in isolation, the ICP immediately initiates isolation.

The analysis revealed the following intrinsic problem domains: Since reports of infectious diseases are generated every afternoon, even on weekends, and every Friday afternoon respectively, an ICP does only recognize infections the morning after the report has been generated. However, these reports are triggering the execution of the second part of the IC process. Hence, they are critical in time.

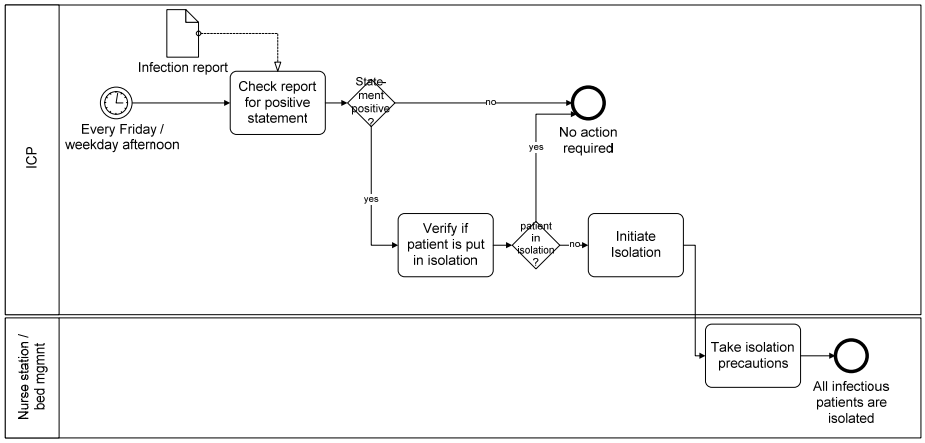

Fig. 2. As-is follow-up process.

The analysis of the IC process clearly revealed that no IT is used after the ICP has received infection reports. In addition, further investigations indicated that ICP did not have any access to the WfMS yet. The review of reports and patient charts needs to be done manually as infection reports are printed and corresponding patient charts are not at hand instantly. A sample inquiry performed in collaboration with ICP indicated that screening all necessary documents takes almost 30 % of ICP's daily work time.

Even though ICP have responsibility for the handling infectious diseases, they are not directly participating in the IC process. Further inquiry revealed that the former process was to leave a voicemail for the ICP, assigned to the nurse station the patient was located at, as soon as an infection disease was stated. Once the ICP received the voicemail, the isolation of infectious patients was initiated and controlled by the ICP. However, since ICP do not work nights or at weekends this process was changed to directly call the floor and notify ICP only through reports. In doing so, the customer reported faster turnaround times in putting patients in isolation even though notification of ICP was delayed, i.e. follow-up processes start delayed.

4.3 To-be Analysis

It became obvious that no change in matters of personnel capacities could be made. Neither could the involvement of ICP in notification tasks be increased nor could the controlling responsibilities of ICP transferred to an IS due to legal prerequisites [26]. The laboratory will still have to notify the nurse station directly, as ICP will, yet, not work during night hours or on weekends. In addition, it was agreed to not work on further improving the turnaround time for putting patients in isolation, but on the elimination of time wasted while generating, delivering, and reading reports as well as screening patient charts. In doing so, it was agreed to optimize IC tasks done by ICP without restructuring the organization or heavily affecting existing clinical and business processes.

Main focus was put on synchronizing the infection notification and the follow-up fragments of the as-is IC process (cf. Figures 1 and 2). The suggestion was to introduce a new workflow supported IC process. It was decided that a workflow should be used to screen new and modified lab results for statements of infectious diseases. Thereby, the lab system and the WfMS have been integrated in a way that every time a lab result is completed in the lab system, data is transferred. In doing so this data is available to users of the system in an instant. For a proper implementation of the integration events defined by Health Level 7 standards have been used. These events are incorporated in the workflow, which evaluates new and modified results and, furthermore, notifies ICP in case a positive infection statement is found. As the case study was a pilot project, it was agreed to limit the workflow to only cover two infections: clostridium difficile (CDIFF) and vancomycin resistant enterococcus (VRE). The workflow can be easily extended to cover other HAI, once clinical conditions have been identified and medical rules are created for an automated evaluation of those conditions.

The uppermost part of Figure 3 illustrates the notification process of the to-be IC process. According to the agreement made with the customer, this part of the IC process was not changed at all. However, the linkage between the notification process and the infection follow-up is made explicit in the to-be process models. A message interface has been added to the notification process and to the follow-up process respectively. The follow-up process is started immediately after notification process and valuable process time is saved by synchronizing both processes parts.

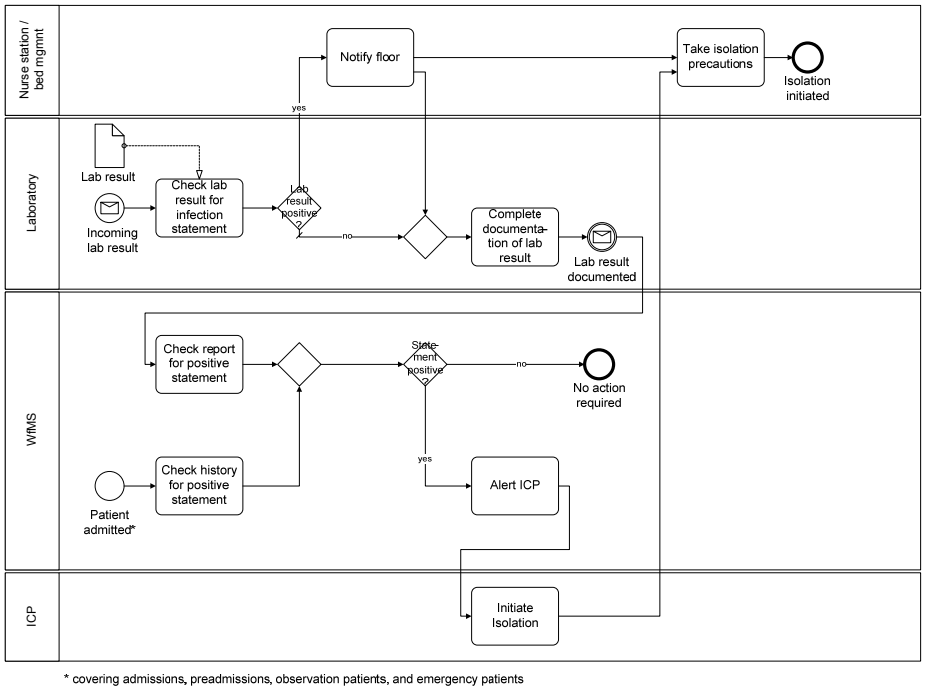

Fig. 3. To-be notification process and to-be follow-up process.

More significant changes have been made to the infection follow-up process that is shown in the lowermost part of Figure 3. The process has been streamlined in order to allow efficient automation through the use of a workflow. To achieve the requested level of automation a WfMS has been introduced to the process. The follow-up process is triggered every time a lab result is documented in the lab system. The lab result is immediately evaluated through workflow. Only if the lab result either indicates a positive CDIFF or VRE statement, the ICP assigned to the nurse station the infectious patient is located at, will be notified. The notification, once again, is performed automatically through workflow. Hence, the ICP will instantly see a new task on his work list. The validation whether a patient has been put in isolation still needs to be done manually, but ICP are now able to access medical records electronically. This enables ICP to work independent of any paper reports or patient charts. Unfortunately, initiating the isolation of patients needs to be executed and monitored manually due to missing integration of participating actors (e.g., bed management).

In consequence of extensively implementing the IC process with the use of IS, new possibilities for further process enhancement have been established. Utilizing new benefits, the to-be infection follow-up process was designed to be executed not only every time a lab result has been documented but also every time a patient is admitted as inpatient, an inpatient is pre-admitted, an outpatient is kept in hospital for observation or a patient requires emergency care. In either case, the last six month of the patient's medical record are screened for an occurrence of an infection. If an infection statement was found within the past six months, the patient is considered to require isolation and infection precautions are taken. Since many HAI (e.g. VRE), are likely to reappear, if the last infection is more recent than six month, these new characteristics enabled the increase of process quality and patient safety in addition to the increase of efficiency.

4.4 Implementation and Controlling

The IC workflow is implemented based on a hierarchical model of procedures and sub-procedures. Thereby, the top level procedure is used to coordinate the overall process flow. The functionality to initiate new workflow instances (e.g., inpatient admit) and terminate existing instances (e.g., patient discharge) is built upon the workflow event handler. So called subscriptions which are basically rules that filter and evaluate selected events allow the definition of case generation and case termination respectively. The workflow has been implemented according to the to-be process models. Thus, a workflow case is initiated the first time a patient is admitted, pre-admitted, put in an observation bed or in the emergency department. The workflow instance will terminate as soon as the patient is sent home (cf. Figure 4).

The purpose of the IC model workflow is to alert users of patients having infectious diseases (e.g., CDIFF, VRE). Therefore, the workflow is designed to immediately check new and modified lab results for patterns indicating a positive statement of an infectious disease. In addition, the workflow checks the medical record for a history of an infectious disease. Alerts on the work lists are created and patients are put on the census list of selected users (e.g., ICP), if a patients has an

active indication to be put in isolation. According to the to-be process the IC workflow is required and triggered by multiple events.

Since every event provides a different set of data, the first three activities deal with consolidating data. This ensures that all workflow relevant data is available instantly. The sub-procedures *PT discharge* and *PT discharge cancelled* are used to perform a delayed workflow termination, if the patient was discharged and the discharge was not cancelled within eight hours.

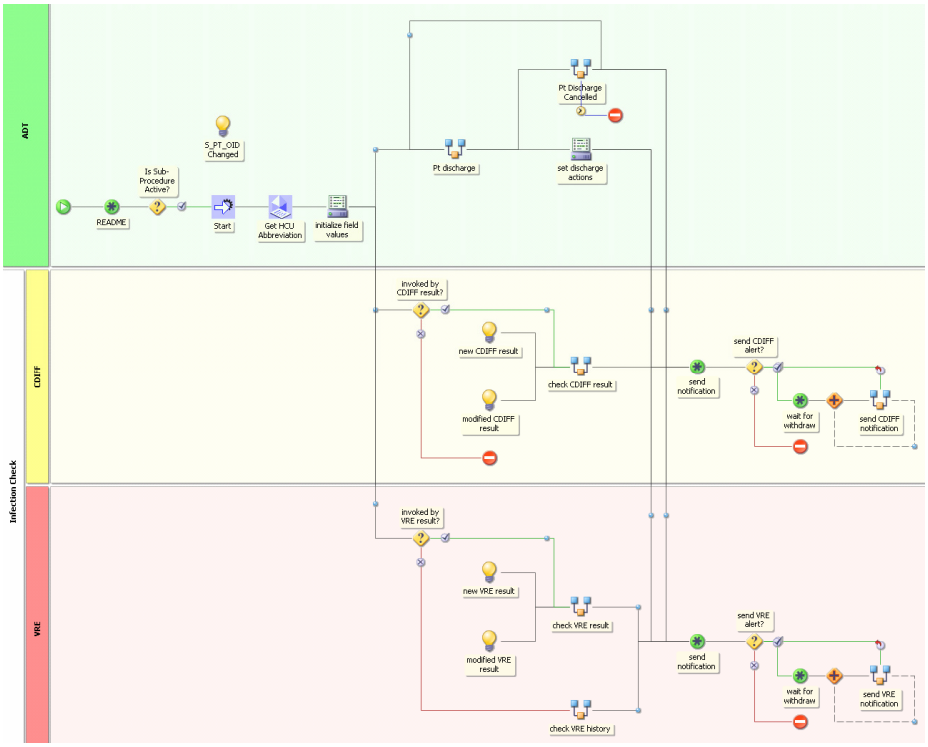

Fig. 4. Infection Control main procedure.

Following the path to the VRE infection checking activities, a conditional router (*invoked by VRE result?*) is used to evaluate whether a lab result needs to be checked for VRE patterns or a patients medical record needs to be screened for positive VRE statements. Both, the checking of a lab result and history screen are performed in sub-procedures. Thus, relevant data like patient identifiers, visit identifiers or result values must be passed to the called sub-procedure. Values returned by the sub-procedure must be mapped in the calling procedure. Once the VRE lab result has been evaluated automatically or the medical records has been screened for previous infections without human intervention, another conditional router is used to examine if a user needs to be alerted and if this patient needs to be added to the user's census list. Alerting users and adding the patient to the census is automatically performed by another sub-procedure call.

Since the notification sub-procedure stays active until the alerted user confirms to have recognized the infection alert (e.g., user releases the alert) and new lab results may be submitted in between, there are cases where the alert message needs to be changed or the alert needs to be withdrawn entirely due to new information provided by new lab results. Therefore, functionality to remove or replace alerts is provided. This is done by withdrawing created alerts (e.g., withdraw connection of *send VRE alert?* and *send VRE* notification) before creating a new alert (*wait for withdraw*). Every time a patient is discharged from hospital all alerts will be removed and the patient will be dropped of the user's census list. If a discharge is cancelled the status of the last notification will be restored.

Concurrent to the project realization data was collected for a detailed analysis of the project outcome. The key metric defined in collaboration with the customer is *time to notification*. Time to notification as measurement of time, represents the time spent notifying an ICP of a newly identified infectious patient. In doing so, time measurement started as soon as a positive lab result had been documented by the laboratory. Time measurement stopped once an ICP had been notified. Data ascertainment of notification dates needed to be done manually. Before the implementation of the IC workflow, ICP were asked to write down dates as soon as they had been notified of infectious patients. After the workflow had been implemented the date of notification was considered to be the date when an ICP released the automatically created infection alert.

Independent of the measurements of time to notification more data was collected in order to identify how much time was spent while screening paper reports or patient charts. Therefore, ICP were asked to write down hours spent on reading reports and screening patient charts before the implementation of the workflow. 95 cases were recorded over the course of two month each for pre- and post-measurement.

The comparison of pre- and post-metrics reveals that time to notification was reduced by more than 75 % after the workflow had been implemented (cf. Table 1).

Table 1. Comparison of post- and pre-measurements.

	Average (VRE)	Average (CDIFF)	Overall Average
Time to notification: pre-metrics	150.00 h	25.68 h	69.68 h
Time to notification: post-metrics	20.67 h	15.68 h	16.99 h
Difference	129.33 h (86.22 %)	10.00 h (38.94 %)	52.69 h (75.61 %)

Time to notification averaged out at 70 hours before the implementation and has reduced to an average of 17 hours after the implementation of the IC workflow. Although 17 hours still appears to be quite a lot, it is obvious that substituting the old paper reports based IC process for the new workflow supported IC process increased efficiency greatly. Furthermore, ICP spent an average of 30 % of their daily work time on screening infection reports and patient charts. This was decreased to almost zero after the implementation of the workflow, since required patient information is now available instantly through the integration of ICP to the WfMS. It has to be noted that before, VRE reports have only been created every Friday afternoon. This means,

that only those VRE occurrences of past Saturday to Friday would appear on Friday's report, which is read on the following Monday at earliest.

4 Conclusion and Next Steps

In this paper we presented reasons for the inappropriateness of greenfield project approaches for the optimization of clinical processes in healthcare.

The main restrictions, which hampered reengineering, originated from judicial and budgetary restrictions. Due to budgetary reasons, it was not possible to increase staff for weekend support or to include all departments (e.g., bed management). Due to judicial restriction, pagers and e-mail were not available at the time because of increased IT test requirements. Furthermore, it was not intended to touch existing IS (e.g., laboratory IS). Several laws or certifications required certain procedures. Significant restrictions impose the certification requirements of The Joint Commission (JCAHO) [20] as it entails implementing the hygiene guidelines of the Center for Disease Control and Prevention (CDC) and the Public Health Service Act [24]. As a consequence, it was, e.g., not possible to transfer authority for ICP to an IS.

Still, significant potential for automating coordination and evaluation task (e.g., calling floors, screening charts) was discovered, utilized and in consequence contributed to patient safety without radically redesigning the organization. Improvements were made through the implementation of HAI history screening, as every inpatient, now, is screened for a positive history. The process quality has been improved through reducing the risk of human errors, since ICP no longer rely on manually generated paper reports or voicemails. The implementation of the workflow greatly contributed to the process of getting health workers, especially ICP, online. It was observed that the automated IC process was functioning as incentive to overcome the negative attitude some health workers might have concerning IT in inpatient care.

References

[1] Becker, J., Kugeler, M., Rosemann, M. (eds.): Process Management: A Guide for the Design of Business Processes. 2nd edn. Springer, Berlin (2007) to appear

[2] Becker, J., zur Mühlen, M.: Towards a Classification Framework for Application Granularity in Workflow Management Systems. In: Proc. 11th International Conference on Advanced Information Systems Engineering (CAiSE). Lecture Notes in Computer Science 1626 (1999) 411-416

[3] Borst, F., Appel, R., Baud, R., Ligier, Y., Scherrer, J. R.: Happy Birthday DIOGENE: A Hospital Information System Born 20 Years Ago. International Journal of Medical Informatics 54 (1999) 157-167

[4] Burke, J. P.: Infection Control: A Problem for Patient Safety. New England Journal of Medicine 348 (2003) 651-656

[5] Centers for Disease Control and Prevention (CDC): National Nosocomial Infections Surveillance (NNIS) System Report, Data Summary from January 1992 through June 2004, Issued October 2004. American Journal of Infection Control 32 (2004) 470-485

[6] Dewire, D. T.: Application Service Providers. Information Systems Management 17 (2000) 14-19

[7] DiMasi, J. A., Hansen, R. W., Grabowski, H. G.: The Price of Innovation: New Estimates of Drug Development Costs. Journal of Health Economics 22 (2003) 151-185
[8] Hammer, M., Champy, J.: Reengineering the Corporation: A Manifesto for Business Revolution. 1st edn. HarperBusiness, New York, NY (1993)
[9] Harmon, P.: Business Process Change: A Manager's Guide to Improving, Redesigning, and Automating Processes. Morgan Kaufmann, San Francisco, CA (2003)
[10] Institute of Medicine: Crossing the quality chasm: A New Health System for the 21st Century. National Academies Press, Washington, DC (2001)
[11] Kohn, L. T., Corrigan, J. M., Donaldson, M. S. (eds.): To Err is Human: Building a Safer Health System. National Academy Press, Washington, DC (2000)
[12] Lenz, R., Reichert, M.: IT Support for Healthcare Processes. In: Proc. 3rd International Conference on Business Process Management (BPM). Lecture Notes in Computer Science 3649 (2005) 354-363
[13] Magruder, C., Burke, M., Hann, N. E., Ludovic, J. A.: Using Information Technology to Improve the Public Health System. Journal of Public Health Management and Practice 11 (2005) 123-130
[14] Object Management Group: Business Process Modeling Notation (BPMN) Specification 1.0 (2006). Available: http://www.omg.org/cgi-bin/apps/doc?dtc/06-02-01.pdf
[15] Panzarasa, S., Stefanelli, M.: Workflow Management Systems for Guideline Implementation. Neurological Sciences 27 (2006) 245-249
[16] Porter, M. E.: Competitive Advantage: Creating and Sustaining Superior Performance. The Free Press, New York, NY (1985)
[17] Pryor, T. A., Hripcsak, G.: The Arden Syntax for Medical Logic Modules. International Journal of Clinical Monitoring and Computing 10 (1993) 215-224
[18] Siemens AG: Soarian® (2007). Available: http://www.soarian.com/
[19] Tao, L.: Shifting Paradigms with the Application Service Provider Model. IEEE Computer 34 (2001) 32-39
[20] The Joint Commission: 2007 National Patient Safety Goals (2007). Available: http://www.jointcommission.org/PatientSafety/NationalPatientSafetyGoals/07_hap_cah_npsgs.htm
[21] The Medical Letter Inc.: Choice of Antibacterial Drugs. Treatment Guidelines from The Medical Letter 2 (2004) 13-26
[22] The Medical Letter Inc.: Treatment of Clostridium Difficile-Associated Disease (CDAD). The Medical Letter on Drugs and Therapeutics 48 (2006) 89-90
[23] TIBCO Software GmbH: TIBCO® Staffware Process Suite (2005). Available: http://bpmleader.tibco.com/de/docs/staffware_processsuite_datasheet.pdf
[24] U.S. Food and Drug Administration: Public Health Service Act (1944). Available: http://www.fda.gov/opacom/laws/phsvcact/phsvcact.htm
[25] van der Aalst, W. M. P., ter Hofstede, A. H. M., Weske, M.: Business Process Management: A Survey. In: Proc. 1st International Conference on Business Process Management (BPM). Lecture Notes in Computre Science 2678 (2003) 1-12
[26] Weber, S. G., Huang, S. S., Oriola, S., Huskins, W. C., Noskin, G. A., Harriman, K., Olmsted, R. N., Bonten, M., Lundstrom, T., Climo, M. W., Roghmann, M.-C., Murphy, C. L., Karchmer, T. B.: Legislative Mandates for Use of Active Surveillance Cultures to Screen for Methicillin-Resistant Staphylococcus Aureus and Vancomycin-Resistant Enterococci: Position Statement From the Joint SHEA and APIC Task Force. Infection Control and Hospital Epidemiology 28 (2007) 249-260
[27] Workflow Managment Coalition: Terminology & Glossary 3.0 (1999). Available: http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf
[28] zur Mühlen, M.: Workflow-based Process Controlling: Foundation, Design and Application of Workflow-driven Process Information Systems. Dissertation. Logos, Berlin (2004)

Declarative and Procedural Approaches for Modelling Clinical Guidelines: Addressing Flexibility Issues

Nataliya Mulyar[1], Maja Pesic[1], Wil M.P. van der Aalst[1] and Mor Peleg[2]

[1] Eindhoven University of Technology
GPO Box 513, NL5600 MB Eindhoven, The Netherlands
{n.mulyar, m.pesic, w.m.p.v.d.aalst}@tue.nl
[2] Department of Management Information Systems, University of Haifa
Mount Carmel, 31905, Israel
{morpeleg}@mis.hevra.haifa.ac.il

Abstract. Recent analysis of clinical Computer-Interpretable Guideline (CIG) modelling languages from the perspective of the control-flow patterns has revealed limited capabilities of these languages to provide flexibility for encoding and executing clinical guidelines. The concept of flexibility is of major importance in the medical-care domain since no guarantee can be given on predicting the state of patients at the point of care. In this paper, we illustrate how the flexibility of CIG modelling languages can be improved by describing clinical guidelines using a *declarative* approach. We propose a CIGDec language for modelling and enacting clinical guidelines.

Keywords: Clinical guidelines, Computer-interpretable guidelines, flexibility, modelling languages, declarative model specification,temporal logic.

1 Introduction

Clinical Computer-Interpretable Guidelines (CIG) impact clinician behavior (i.e., quality of patient care, costs, etc.) to a great extent when they are implemented as computerized guidelines that deliver patient-specific recommendations during patient encounters [1]. A number of guideline modelling languages have been developed to represent guidelines in a machine and human understandable format that enables guideline execution. Control-flow perspective of guidelines significantly influences the clinical behavior, because it determines the order of actions in medical treatments. Other perspectives (e.g., a model of patient data including temporal data, a model of medical actions and decisions, etc.) add more contextual details to the control-flow perspective, determining the exact favorable clinical behavior. Unfortunately, due to the absence of a single standard for developing CIG modelling languages, the functionality of decision-support systems employing such modelling languages from the perspective of the control-flow differs to a great extent.

We analyzed the suitability of four modelling languages Asbru, PRO*forma*, GLIF and EON for expressing control-flow patterns [2] and revealed that these languages do not offer more control-flow flexibility than process modelling languages employed by the Workflow Management Systems (WFMS) [3]. This is remarkable since one would expect CIG modelling-languages to offer dedicated constructs allowing for more build-time and runtime flexibility. Accommodating *flexibility* into guidelines means that the

CIG would be sensitive to the characteristics of specific patients and specific health care organizations [4].

The modelling languages we analyzed explicitly model a care process by specifying the steps and the order in which these steps are to be executed. Although process languages allow for some flexibility by means of modelling alternative paths, any of which could be taken depending on some a-priori available data, they are incapable of handling exceptional or unpredicted situations. Exceptional situations have to be modelled explicitly. However, modelling of all possible scenarios results in complex models and is not feasible since exceptional situations and emergencies may arise at any point in time. This makes it difficult or even impossible to oversee what activity should be performed next. To overcome these problems, i.e. reduce the complexity of models, and to allow for more flexibility in selecting an execution path, *in this paper* we propose CIGDec as a declarative language for modelling clinical guidelines. Unlike imperative languages, declarative languages specify the "what" task should be performed without determining of the "how" to perform it. CIGDec specifies by means of constraints the rules that should be adhered to by a user during a process execution while leaving a lot of freedom to the user in selecting tasks and defining the order in which they can be executed. CIGDec can be considered as a variant of ConDec [5] and DecSerFlow [6].

The remainder of this paper is organized as follows. In Section 2 we introduce CIG modelling languages Asrbu, GLIF, EON and PRO*forma* using a patient-diagnosis scenario. In Section 3 we introduce CIGDec and illustrate a CIGDec model of the patient-diagnosis scenario. We discuss the drawbacks and advantages of the proposed language in Section 4. Related work is presented in Section 5. Section 6 concludes the paper.

2 Computer-Interpretable Guidelines

This section describes the main concepts of four well-known CIG modelling languages: Asbru, EON, GLIF, and PRO*forma*. These have been evaluated from the control-flow perspective using the workflow patterns [7]. We introduce the main concepts of these languages by modelling the following patient diagnosis scenario in the tools AsbruView, Protege-2000 (for EON and GLIF) and Tallis respectively. A patient is registered at a hospital, after which he is consulted by a doctor. The doctor directs the patient to pass a blood test and urine test. When the results of both tests become available, the doctor sets the diagnosis and defines the treatment strategy.

While specifying the behavior of the scenario, we immediately reflect on the possibilities to deviate from this scenario required for example in an emergency case. In particular, we indicate whether it is possible to skip a patient registration step and immediately start with the diagnosis; to perform multiple tests of the same kind or perform only one of them; and to perform the consultancy by the doctor after performing one of the tests again. Next to it, we indicate the degree of support of the control-flow patterns by the analyzed modelling languages. Table 1 summarizes the comparison of the CIG modelling languages from the perspective of the control-flow patterns [7]. The complete description of the patterns and how they are supported by the analyzed languages can be found in [8, 3] respectively.

Figure 1 presents the scenario modelled in AsbruView, which is a markup tool developed to support authoring of guidelines in Asbru [9]. A process model in Asbru

Basic Control–flow	1	2	3	4	New Patterns	1	2	3	4
1. Sequence	+	+	+	+	21. Structured Loop	+	+	+	+
2. Parallel Split	+	+	+	+	22. Recursion	+	-	-	-
3. Synchronization	+	+	+	+	23. Transient Trigger	-	-	-	+
4. Exclusive Choice	+	+	+	+	24. Persistent Trigger	-	-	+	+
5. Simple Merge	+	+	+	+	25. Cancel Region	-	-	-	-
Advanced Branching and Synchronization					26. Cancel Multiple Instance Activity	+	-	+	+
6. Multi-choice	+	+	+	+	27. Complete Multiple Instance Activity	+	-	-	+
7. Structured Synchronizing Merge	+/-	-	-	+	28. Blocking Discriminator	-	-	-	-
8. Multi-merge	-	-	-	-	29. Cancelling Discriminator	+	-	-	+
9. Structured Discriminator	+	+	+	+	30. Structured N-out-of-M Join	+	-	+	+
Structural Patterns					31. Blocking N-out-of-M Join	-	-	-	-
10. Arbitrary Cycles	-	+	+	-	32. Cancelling N-out-of-M Join	-	-	-	+
11. Implicit Termination	+	+	+	+	33. Generalized AND-Join	-	-	-	-
Multiple Instances Patterns					34. Static N-out-of-M Join for MIs	-	-	-	-
12. MI without Synchronization	-	-	-	-	35. Static N-out-of-M Join for MIs with Canc.	-	-	-	-
13. MI with a priori Design Time Knowledge	+/-	+/-	+/-	+/-	36. Dynamic N-out-of-M Join for MIs	-	-	-	-
14. MI with a priori Run-Time Knowledge	-	-	-	-	37. Acyclic Synchronizing Merge	-	-	-	+
15. MI without a priori Run-Time Knowledge	-	-	-	-	38. General Synchronizing Merge	-	-	-	-
State-Based Patterns					39. Critical Section	+	-	+	-
16. Deferred Choice	+	-	+	+	40. Interleaved Routing	+	-	+	-
17. Interleaved Parallel Routing	+	-	-	-	41. Thread Merge	-	-	-	-
18. Milestone	-	-	-	+	42. Thread Split	-	-	-	-
Cancellation Patterns					43. Explicit Termination	-	-	-	-
19. Cancel Activity	+	+	+	+					
20. Cancel Case	+	-	+/-	+					

Table 1. Support for the Control–flow Patterns in (1)Asbru, (2)EON, (3)GLIF, and (4)PRO*forma*

[10] is represented by means of a time-oriented skeletal plan. Skeletal plans are plan schemata at various levels of detail, which capture the essence of the procedure, but leave room for execution-time flexibility. The root plan A is composed of a set of other plans that are represented as 3-dimensional objects, where the width represents the time axis, the depth represents plans on the same level of the decomposition (i.e. which are performed in parallel), and the height represents the decomposition of plans into sub-plans.

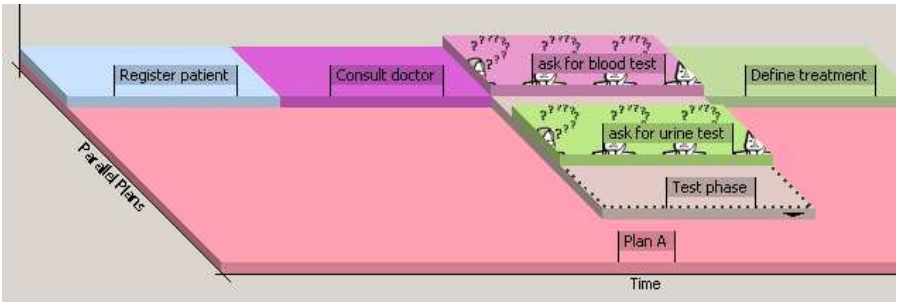

Fig. 1. The patient-diagnosis scenario modelled in Asbru

As the time axis shows, plans *Register patient*, *Consult with doctor*, *Test phase* and *Define the treatment* are executed sequentially. The *Test phase* is a parallel plan consisting of two activities *ask for urine test* and *ask for blood test*. The parallel plan requires all enclosed plans to be completed in order to pass the flow of control to the next plan. In this model, only two types of plans were used: sequential (root plan) and parallel plan (Test phase plan). In AbsruView plans of type: Any-order, Unordered, Cyclical, and If-then-else, and actions of type: Ask and Variable Assignment can be visualized.

Deviations from the modelled scenario are not possible in AsbruView, since all plans are structured and their order is strictly defined. It would be possible to adjust the model and implicitly incorporate all required execution paths. In particular, the Cyclical Plan should be used in order to iterate the execution of a certain task. In order to relax the parallel order of the blood- and urine-tests' tasks, an Any-order Plan could be used. However, the behavior of the model would be still deterministic and not allow for much flexibility. In Asbru there is a concept of plan activation mode. It allows conditions for aborting, suspending and resuming a plan. This can be relevant for the case of registering a patient and not having all the needed data initially: a plan is suspended and later resumed. As the pattern-based analysis showed [3], Asbru is able to support 20 out of 43 control-flow patterns. Asbru uniquely supports the recursive calls and interleaved parallel routing, which are the features not directly supported by other analyzed languages.

Main modelling entities in EON [11] are scenarios, action steps, branching, decisions, and synchronization [12, 13]. A scenario is used to characterize the state of a patient. There are two types of Decision steps in EON, i.e. a Case step (select precisely one branch) and a Choice step (select at least one branch). An Action step is used to specify a set of action specifications or a sub-guideline that are to be carried out. Branch and Synchronization steps are used to specify parallel execution. We omit EON model since it is very similar to model created in GLIF (see Fig. 2).

The following features offered by EON can be used in order to make the model of the patient-diagnosis scenario more flexible. A Scenario can be used to model different entry points to the model. This allows to "jump" into the middle of the model and to start execution from that point. This feature is useful for emergency cases where for example a registration step has to be skipped and immediate treatment procedure has to be started. Unfortunately, EON offers not much flexibility with respect to synchronization of multiple branches, i.e. it allows the *define treatment* task to be executed only if a single or all branches have been executed. However, it is incapable of predicting how many branches were selected and performing a partial synchronization after all selected branches were executed. From all analyzed modelling languages, EON supports the lowest number of the control-flow patterns, i.e. only 11 out of 43.

GLIF3.5 [14] is a specification method for structured representation of guidelines. To create a model in GLIF, an ontology schema and a graph widget have to be loaded into the *Protege-2000* environment. Figure 2(a) visualizes the GLIF model of the basic patient-diagnosis scenario. In GLIF3.5 five main modelling entities are used for process modelling, i.e. an Action Step, a Branch Step, a Decision Step, a Patient-State Step, and a Synchronization Step. An Action Step is a block for specifying a set of tasks to be performed, without constraints set on the execution order. It allows for including sub-guidelines into the model. Decision steps are used for conditional and unconditional routing of the flow to one out of multiple steps. Branch and Synchronization steps are used for modelling concurrent steps. A Patient-State Step is a guideline step used for describing a patient state and for specifying an entry point(s) to a guideline.

In order to allow the behavior of the basic patient-diagnosis scenario shown in Figure 2(a) to deviate, all possible pathes have to be explicitly modelled. Figure 2(b) represents a scenario, in which *Register patient* step can be done in parallel to any other

step, but it has to be exactly once to complete the process (if more than once is desired, an iteration condition for *Register patient* step can be added which resembles a while loop: while not all patient data has been entered, repeat Register Patient. In this scenario, a decision can be taken to order tests or to proceed to treatment without tests. However, treatment or ordering of tests cannot be done before consulting with a doctor. One or two tests can be ordered before proceeding to treatment. Figure 2(b) shows how complex the model has become after we introduced several deviations from the basic scenario. Thus, this specification needs to model graphically all the possible paths of execution, and it is not very scalable.

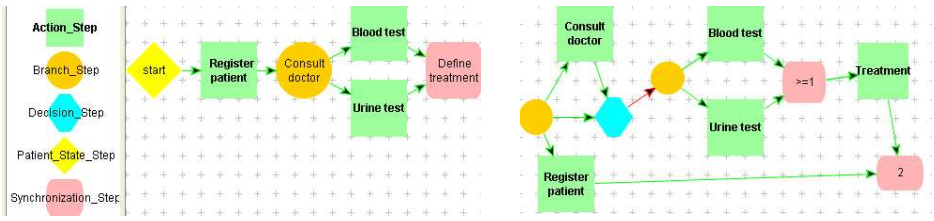

Fig. 2. The patient-diagnosis scenario modelled in GLIF3.5/*Protege*

Similar to EON, GLIF allows multiple entry points into the model to be specified by means of the Patient-State step. This allows the execution to start from any point where a patient enters a scenario model while skipping tasks-predecessors. GLIF offers more variants for synchronizing parallel branches, i.e. to synchronize after one, several or all tasks have been completed. However, GLIF is incapable of synchronizing branches when it is unknown which branches and how many of them will be chosen. This explains why the number of control-flow patterns supported by GLIF (17 out of 43) is bigger than in EON but still smaller than Asbru.

PRO*forma* [15] is a formal knowledge representation language for authoring, publishing and executing clinical guidelines. It deliberately supports a minimal set of modelling constructs: actions, compound plans, decisions, and enquiries that can be used as tasks in a task network. In addition, a keystone may be used to denote a generic task in a task network. All tasks share attributes describing goals, control flow, preconditions, and postconditions. A model of the basic patient-diagnosis scenario created in Tallis is shown in Figure 3(a). Note that in PRO*forma* control-flow behavior is captured by modelling constructs in combination with the scheduling constraints. Scheduling constraints are visualized as arrows connecting two tasks, meaning that the task at the tail of the arrow may become enabled only after the task at the head of the arrow has completed. To deviate from the basic scenario, some of the scheduling constraints should be removed as it is shown in Figure 3(b).

In contrast to all examined languages, PRO*forma* allows for late modelling, i.e. if it is not clear in advance what steps exactly should be performed, these steps are modelled by means of keystones, which are substituted by a desired type of the task before the model is deployed. Furthermore, by means of triggers it is possible to specify that a task has to be performed even if the task's preconditions were not satisfied. PRO*forma* also

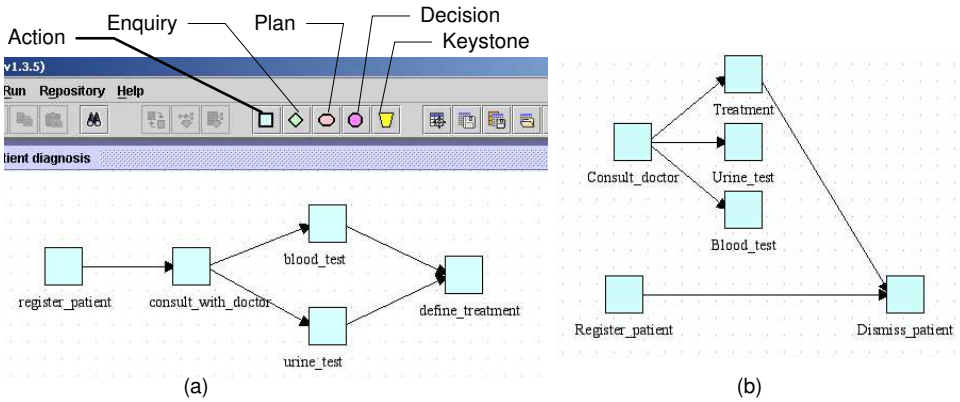

Fig. 3. The patient-diagnosis scenario modelled in PRO*forma*/Tallis

allows for more flexibility during the synchronization of multiple paths, thus it is able to predict how many paths from the available ones were selected and to merge them when they have completed. Furthermore, scheduling constraints in PRO*forma* are not obligatory. This means that stand-alone tasks may be activated upon the fulfillment of a pre-condition. PRO*forma* has the highest degree of pattern-support from all analyzed languages, i.e. it supports 22 out of 43 patterns.

The medical community has always emphasized that it is impossible to use workflow formalisms because of specific requirements such as flexibility. However, when we examined guideline modelling languages we didn't find more flexibility than in todays workflow and BPM products. Given a large variety of process modelling languages nowadays it makes no sense to develop more complicated language which would support more control-flow patterns. Instead, we take a completely new approach and propose a CIGDec language for encoding clinical guidelines.

3 Declarative description of clinical guidelines

In this section we present the CIGDec declarative language and show benefits of applying it for modeling clinical guidelines.

Modelers who use traditional CIG modelling languages have to represent all possible scenarios (normal and exceptional) that can occur during the execution. Such a model has to include all possible scenarios that can occur during the execution. This means that CIG modelers have to predict in detail all possible execution paths in advance for the guideline they are modelling. The model itself tends to be very complex and strictly predefines all relationships between all steps in the guideline. Such a model not only prescribes to users what to do, but it also contains a detailed specification about how to do it. Hence, traditional CIG modelling languages are of an imperative nature.

CIGDec is a declarative language, i.e., its models specify what to do and leave it up to the user to decide how to work depending on the case. CIGDec models do not require all possible scenarios to be predicted in advance. On the contrary, the model consists of a set of tasks and some dependencies (relationships) between these tasks.

Dependencies between tasks can be seen as some general rules that should always hold in the guideline. Any task in the model can be performed by a user if and only if none of the specified rules is violated. As an extreme example, a CIGDec model that consists only of a set of tasks without dependencies would represent a completely free guideline, where a user can execute any task in any desired order. As more rules in the model as less possibilities to deviate from a certain execution order is given to the user. Therefore, rules constrain the model. Hence, we refer to dependencies between tasks (rules) as to *constraints*.

Any CIG model consists of a set of tasks and some relationships between them specifying the exact order of tasks. Typically, traditional languages use a predefined set of constructs that can be used to define relations between tasks: 1) sequence, 2) choice, 3) parallelism, and 4) iteration. These constructs are used to define the exact control-flow (order of tasks) in the guideline. In CIGDec, this set of constructs is unlimited, i.e., constructs can be added, changed and removed, depending on the requirements of the application, domain, users, etc. We refer to constructs used for defining possible types of dependencies between tasks in CIGDec as to *constraint templates*. Each template has its semantics, which is formally represented by one Linear Temporal Logic (LTL) formula. This semantics is used for the computerized enactment of the guideline [5]. LTL is extensively used in the field of model checking, where the target model is verified against properties specified in LTL [16, 17]. For computerized enactment of CIGDec model we use algorithms for translating LTL expressions into automata developed in the model checking field [18, 6, 5]. Since LTL formulas can be very complex and hard to understand, each template also has unique graphical representation for users. In this way, we ensure that CIGDec users do not have to be LTL experts in order to work with models [5]. Although the set of templates is 'open', we propose a starting collection of eleven templates in [19].

When looking at a traditional CIG model, one usually tries to find the starting point and then follows the control-flow until the end point is reached. This cannot be applied to CIGDec models. Constructs (lines) do not necessarily describe the order of tasks, but rather various dependencies between them. In our starting set of constraint templates we distinguish between two types of templates: 'existence' (unary) templates, and binary templates that can represent a 'relation' or 'negative relation'.

'Existence' templates are unary templates because they involve only one task. Generally, they define the cardinality (possible number of executions) of the task. Binary templates involve two tasks. For example, a special line between tasks might mean that these two tasks include each other (e.g., 'co-existence' template between tasks *A* and *B* specifies that if *A* happens then *B* happens and vice versa, without specifying in which order). The 'responded existence' constraint specifies that if one task is performed then the other task before or after the first one. There are also some binary templates that consider the order of activities. One example is the 'response' template, which specifies that the a given task has to be performed at least once after the other task has been completed. Note that in all these examples it was possible to have an arbitrary execution of other tasks between the two related tasks.

3.1 CIGDec model for the diagnosis scenario

Figure 4 depicts a CIGDec model of our patient-diagnosis scenario. It consists of five tasks. In an extreme case, it would be possible to make and use the model consisting only out of these tasks and without any constraints. This would be a unrestricted model allowing for maximum flexibility, where tasks could be executed an arbitrary number of times ('0..*') and in any order. This model would have an infinite number of execution possibilities (different process instances). However, to develop a model that provides guidance, we add five constraints derived from three constraint templates.

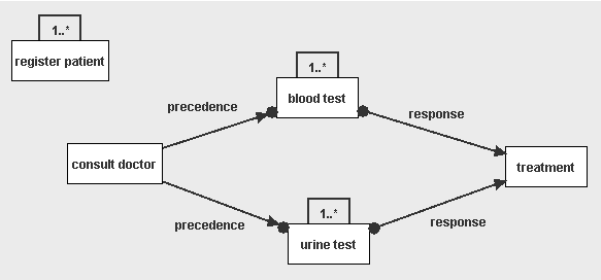

Fig. 4. CIGDec model for the diagnosis scenario.

First, there is one unary (involving one task) constraint created from the template 'existence' - constraint presented as cardinality *1..** above the task *register patient*. This constraint specifies that the task *register patient* has to be executed at least once within one process (guideline) enactment.

Second, there are two constraints created from the template 'precedence' as shown in Figure 4: one between tasks *consult doctor* and *blood test* and one between tasks *consult doctor* and *urine test*. Precedence is a binary template, i.e., it defines a dependency between two tasks. A 'precedence' between two tasks *A* and *B* means that task *B* can only be executed after task *A* was executed at least once [6]. It is possible that other tasks are executed between *A* and *B*. Hence, if we want to execute task *blood test* we can do so only after we have executed task *consult doctor*. Note that other tasks from the model can be executed between *consult doctor* and *blood test*. Task *test urine* also has a 'precedence' relation with task *consult doctor* and it can be executed only after task *consult doctor*. Similarly, there could be other tasks between them. Moreover, the doctor may be consulted multiple times before and after doing the tests.

Third, we use a binary template 'response' to create two constraints: one between tasks *blood test* and *treatment* and one between tasks *urine test* and *treatment*. Template 'response' between tasks *A* and *B* defines that after every execution of task *A* task *B* has to be executed at least once while it is possible that other tasks are executed between *A* and *B*. Thus, after every *blood test* at least one *treatment* should follow, and there could be other tasks from the model executed between them. The same holds for tasks *urine test* and *treatment*.

The possibilities given to a user during execution of the model depicted in Figure 4 are defined as a combination of all five constraints in the model. When looking at the models designed by means of the analyzed language Asbru, the execution always had

to start with the task *register patient*. This may cause problems in cases of emergency, when there is no time for the registration requiring the procedure with doctor (task *consult doctor*) to start immediately. While in EON and GLIF allow multiple entry-points to a scenario, these entrance steps have to be modelled explicitly. In PRO*forma* a task can be modelled without use of scheduling constraints which allows this task to be executed at any moment. Note however, that the CIG languages assume that a task can be executed once during the model execution or *iteratively* a specified number of times. In CIGDec model a patient-registration step can be performed at any moment during the CIGDec process. Furthermore, CIGDec model allows to perform *register patient* multiple times in case the required data is not available on time.

If we look at the traditional models Figures 1, 2 and 3 (i.e. mode using Absru and EON, GLIF and PRO*forma*), task *consult doctor* was executed exactly once. CIGDec model allows this task not to be executed at all, but it also allows it to be executed multiple times. For example, some patients use medication periodically. For them only the *treatment* task has to be performed either before or after the *register patient* has been executed. On the other hand, in some complex cases, task *consult doctor* can be performed more than once at various points during the CIGDec execution.

If necessary, a doctor can order a *blood test* many times or not at all during the CIGDec process. However, constraint 'precedence' between this task and *consult doctor* makes sure that *blood test* can not be done for a patient that has not seen the doctor before. Note that his holds only for the first *blood test*. Sometimes, the results can be unexpected and doctor can order a different type of *blood test* without having to see the patient again. After every *blood test*, task *treatment* is performed. It is possible that during *treatment* no medication is prescribed due to the good test results. However, it is also possible to wait and to perform several *blood tests* in order to make an informed decision before the task *treatment* is performed. Since task *urine test* has the same relationships as task *blood test* ('precedence' with *consult doctor* and 'response' with *treatment*), the same variants of execution paths hold like for the task *blood test*. However, note that none of the tasks 'blood test' and 'urine test' do not have to execute at all, or each of them can be executed one or more times, or only one of them can be executed one or more times.

CIGDec model from Figure 4 could be used to realize the following scenarios. First, in the 'case A' a periodical medication is prescribed to a chronic patient: only *register patient* and *treatment* tasks are executed. In the 'case B' an urgent visit starts directly with *consult doctor* and only afterwards the task *register patient* is executed. The *urine test* was not necessary. The results of the *blood test* were unclear so the *treatment* is executed only after the results of the second *blood test* became available and an additional *consult doctor* task. In the 'case C', the situation was not urgent, so task *register patient* was performed before the task *consult doctor*. Both *urine test* and *blood test* are performed. However, due to alarming results of the *urine test* an immediate *treatment* was executed to prescribe appropriate medication. The results of *blood test* arrived later, and an additional *treatment* task was executed to handle the *blood test* results as well.

4 Discussion

We have shown that CIGDec can be used to define the degree of flexibility given to a user during the process execution. We have also indicated that a degree of the absolute flexibility can be reached by leaving out all constraints resulting in the freedom given to a user to select any task and execute tasks in any desired order. Since the degree of flexibility has to be controlled in the context of medical care in order to adhere to strict and desirable recommendations, the mandatory and optional constraints have to specified for a modelled guideline. To control the adherence to the specified constraints, the execution engine CIGDec prohibits the violation of the mandatory constraints while allowing the optional constraints to be neglected. All user steps that might result in the violation of constraints are communicated to a user by means of warnings.

The advantages of the proposed CIGDec-based approach over the analyzed modelling languages that employ the imperative approach are as follows:

- CIGDec enables the *flexibility in selection*, meaning that a user executing a model specified in CIGDec gets a freedom in choosing an execution sequence, without requiring this sequence to be thought of in advance and explicitly modelled during the design-time.
- CIGDec enables *late binding*, meaning that it allows to choose an appropriate task at the point of care. This feature is particular important in modelling of CIG since it is not always possible to predict what steps will need to be executed, thus the task selection is case-dependent.
- CIGDec ensures the *absence of change*, meaning that it prohibits choices of users that would violate mandatory constraints.
- CIGDec allows for extendability and allows new LTL formulas to be introduced, thus applicability of CIGDec could be tailored to a specific situation.

The disadvantages of using CIGDec are as follows:

- If a process to be modelled has to be very strict and should allow for flexibility, then the use of CIGDec may result in a complex model.
- CIGDec aims at the modelling of rather small processes, since the description of large processes (containing approximately several thousands of tasks) becomes difficult to understand.

Since both imperative and declarative languages have disadvantages, in order to improve the flexibility of the CIG modelling languages we recommend to augment the CIG languages with the features offered by CIGDec.

5 Related Work

The recent *Workflow Patterns initiative* [2] has taken an empirical approach to identifying the most common control constructs inherent to modelling languages adopted by workflow systems. In particular, a broad survey of modelling languages resulted in 20 workflow patterns being identified [7]. In this paper we have used the revised set of the control-flow patterns [8] to evaluate CIS's modelling languages.

There had been many attempts to enrich the flexibility of workflow (process) management systems. Case-handling systems are systems that offer more flexility by focusing on the whole case (process instance), instead of individual tasks [20]. An example of such a system is FLOWer [21], where users can 'move up and down' the process by opening, sipping and re-doing tasks, rather than just executing tasks. Although users have a major influence on execution in FLOWer, their actions are seen as going backwards or forward in a traditional process model. Moreover, this might some unwanted side-effects. For example, if the user wishes to execute again (re-do) an earlier task, s(he) will also have to execute again (re-do) all tasks that followed it. Unlike in FLOWer, deviations are not seen as an exception in CIGDec but as 'normal' behavior while the process instance unfolds further according to the choices of users.

Flexibility of process enactment tools is greatly increased by their adaptivity. ADEPT is an example of an adaptive system where users can change the process model during the enactment [22]. ADEPT is a powerful tool which enables users to insert, move and delete tasks form the process instance they are currently working on. However, the user has to be a process modelling expert in order to change the model. Moreover, in medical domain cases may have many differences and adaptations would be too frequent and time consuming. CIGDec does not see deviations as changes in the model and a good-designed CIGDec model can cover a wide variety of cases.

One of a promising ways to introduce flexibility is to replace imperative by declarative. Various declarative languages "describe the dependency relationships between tasks, rather than procedurally describing sequences of action" [23]. Generally, declarative languages propose modeling constraints that drive the model enactment [23–25]. Constraints describe dependencies between model elements. Constraints are specified using pre and post conditions for target task [25], dependencies between states of tasks (enabled, active, ready, etc.) [23] or various model-related concepts [24].

6 Conclusions

In this paper, we have proposed a declarative approach which could be applied to overcome problems experienced by the imperative languages used for modelling clinical guidelines. In particular, we have shown how by means of applying the CIGDec language more flexibility in selection can be achieved than the considered CIG modelling languages offer. Furthermore, we showed how the model declared in CIGDec can be enacted. In addition, we discussed differences between the proposed declarative and analyzed imperative languages, their advantages and disadvantages, and made a proposition to combine the features of imperative and declarative approaches in order to increase their applicability and usability.

References

1. Peleg, M.: Chapter 4-2: Guideline and Workflow Models. In R.A., G., ed.: Medical Decision Support: Computer-Based Approaches to Improving Healthcare Quality and Safety. Elsevier (2006)
2. Workflow Patterns Home Page. http://www.workflowpatterns.com

3. Mulyar, N., Aalst, W., Peleg, M.: A Pattern-based Analysis of Clinical Computer-Interpretable Guideline Modelling Languages. Technical report, Center Report BPM-06-29, BPMcenter.org (2006)
4. Wald, J., Pedraza, L., Murphy, M., Kuperman, G.: Requirements development for a patient computing system. In: Proc. AMIA Symp. (2001) 731–735
5. Pesic, M., Aalst, W.: A declarative approach for flexible business processes management. In: Business Process Management Workshops. (2006) 169–180
6. Aalst, W., Pesic, M.: Specifying, discovering, and monitoring service flows: Making web services process-aware. BPM Center Report BPM-06-09, BPM Center, BPMcenter.org (2006)
7. Aalst, W., Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow Patterns. Distributed and Parallel Databases **14**(1) (2003) 5–51
8. Russell, N., Hofstede, A., Aalst, W., Mulyar., N.: Workflow Control-Flow Patterns: A Revised View. BPM Center Report BPM-06-22, BPMcenter.org (2006)
9. Shahar, Y., Miksch, S., Johnson, P.: The Asgaard Project: a task-specific Framework for the application and critiquing of time-oriented clinical guidelines. Artif Intell Med (14) (1998) 29–51
10. Seyfang, A., Kosara, R., Miksch, S.: Asbrus Reference Manual, V.7.3. Technical report, Vienna Univ. of Techn., Inst. of SW Techn., Vienna. Report No.: Asgaard-TR-2002-1 (2002)
11. Tu, S., Musen, M.: A flexible approach to guideline modeling. In: Proc AMIA Symp. (1999) 420–424
12. Tu, S., Musen, M.: From guideline Modeling to guideline execution: Defining guideline-based decision-support Services. In: Proc AMIA Annu Symp. (2000) 863–867
13. Tu, S.: The eon guideline model. http://smi.stanford.edu/projects/eon (2006)
14. Boxwala, A., M.Peleg, Tu, S., O.Oqunyemi, Zeng, Q., Wang, D., et al.: GLIF3: a representation format for sharable computer-interpretable clinical practive guidelines. Biomedical Informatics **37**(3) (2004) 147–161
15. Fox, J., Johns, N., Rahmanzadeh, A.: Disseminating Medical Knowledge: The PROforma Approach. Artificial Intelligence in Medicine **14**(1) (1998) 157–182
16. Jr., E.C., Grumberg, O., Peled, D.: Model Checking. The MIT Press, Cambridge, Massachusetts and London, UK (1999)
17. Holzmann, G.: The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley, Boston, Massachusetts, USA (2003)
18. Giannakopoulou, D., Havelund, K.: Automata-based verification of temporal properties on running programs. In: ASE '01: Proc. of the 16th IEEE int. conf. on Automated sw engineering, Washington, DC, USA, IEEE Computer Society (2001) 412
19. Mulyar, N., Pesic, M., Aalst, W., Peleg, M.: Towards the Flexibility in Clinical Guideline Modelling Languages. Technical report, Center Report BPM, BPMcenter.org (2007)
20. Reijers, H., Rigter, J., Aalst, W.: The Case Handling Case. International Journal of Cooperative Information Systems **12**(3) (2003) 365–391
21. Pallas Athena: Flower User Manual. Pallas Athena BV, Apeldoorn, The Netherlands. (2002)
22. Reichert, M., Dadam, P.: ADEPTflex: Supporting Dynamic Changes of Workflow without Losing Control. Journal of Intelligent Information Systems **10**(2) (1998) 93–129
23. Dourish, P., Holmes, J., MacLean, A., Marqvardsen, P., Zbyslaw, A.: Freeflow: mediating between representation and action in workflow systems. In: Proc. of CSCW '96. (1996) 190–198
24. Mangan, P., Sadiq, S.: On building workflow models for flexible processes. In: ADC '02: Proc. of the 13th Australasian database conf., Australian Computer Society, Inc. (2002) 103–109
25. Wainer, J., de Lima Bezerra, F. In: Groupware: Design, Implementation, and Use. Volume 2806. Springer Berlin / Heidelberg (2003) 151 – 158

Managing Socio-Technical Interactions in Healthcare Systems

Osama El-Hassan, José Luiz Fiadeiro and Reiko Heckel

Dept. of Computer Science, University of Leicester, University Road
Leicester, LE1 7RH, United Kingdom
{oehe1, jose, reiko}@mcs.le.ac.uk

Abstract. We put forward an architectural framework that promotes the externalisation of the social dimension that arises in software-intensive systems which, like in healthcare, exhibit interactions between humans (social components) and technical components (devices, computer-based systems, and so on) that are critical for the domain in which they operate. Our framework is based on a new class of architectural connectors (social laws) that provide mechanisms through which the biddability of human interactions can be taken into account and the sub-ideal situations that result from the violation of organisational norms can be modelled and acted upon by reconfiguring the socio-technical systems. Our approach is based on formal, algebraic graph-based representations and transformations.

1 Introduction

One of the main sources of difficulties in healthcare oriented process management is the fact that staff may deviate from prescribed medical or organisational workflows. Such deviations are not "errors" but, rather, result from the fact that situations may arise in which humans may need to interact with machines in "sub-ideal" states [17]:

> *"Medical personnel must be free to react and is trained to do so. Therefore, in addition to the customization of a pathway schema at instance creation time, it must be possible to dynamically adapt in-progress treatment cases (i.e., process instances) during runtime. […] As a consequence the actual patient treatment process may deviate from the original treatment plan and the related medical pathway respectively. However, such deviations from the pre-planned process must not lead to errors or inconsistencies".*

Freedom to (re)act matches perfectly the characteristics of *biddable domains* as identified by Jackson [16]: "*[people] can be joined to adhere to certain behaviour, but may or may not obey the injunction*". The problem, as highlighted in the quote above, is to endow systems with the degree of flexibility that allows them to adapt dynamically to changes from ideal to sub-ideal states.

In this paper, we explore the use of architectural modelling techniques for making systems adaptable through dynamic reconfiguration [2]. Such techniques rely on the

externalisation of the coordination of the interactions among components of the system in explicit connectors [1]. One can then reconfigure the system by changing the connectors without changing the components themselves. However, in order to be able to address interactions in which some of the components are human, as is the case in healthcare, we need new modelling primitives that extend the notion of connector. For instance, we should be able to model the interactions that, in a healthcare system, are associated with the roles performed by medical staff when engaging with fellow staff, software applications or machines.

Our approach addresses what are sometimes called "socio-technical systems": systems that include a "social dimension" in the sense that people (or groups of people) need to be considered not as external users but as another class of components that, together with software and devices, perform roles that are vital for the "good" behaviour of the system. In order to bring human interaction within the boundaries of systems, we need to be able to refer to normative concepts like permissions, obligations and power, and to model the violations that can take place so that processes and underlying software can be reconfigured to react to non-normative situations in ways that ensure agreed, possibly minimal, levels of service. In this paper, we are particularly interested in capturing sub-ideal situations that are caused by violating an obligation or a permission to perform a certain action or activity.

From this discussion, it should be obvious that our approach differs from the perspective on human interaction taken in HCI (Human-Computer Interactions), which addresses user interface issues, or CSCW (Computer Supported Cooperative Work) [], which tackles person-to-person computer-mediated collaborations. Our focus is not on *user* interaction but on peer-to-peer interactions. Our starting point is a combination of recent work on software architecture [2] and the knowledge that has been accumulated in multidisciplinary areas of research such as deontic logic [22], multi-agent systems [10], role-based access control [21] and other social studies. In [8], we made a preliminary proposal for a new class of architectural primitives and formal configuration modelling techniques based on [24]. These configuration primitives allow us to specify when and how to reconfigure the software architecture, at runtime, in a way that reflects the normative positions of humans as engaged in interactions within organisational settings. In this paper, we provide an overview of this work as applied to healthcare and develop in more detail the semantics of the reconfiguration operations.

Section 2 constitutes a brief discussion of the architectural modelling approach and addresses the new primitives, through which biddable interactions can be coordinated, then after. Section 3 provides a mathematical semantics over graph-based representations and transformations followed by an example.

2 The Architectural Approach

2.1 Architectural Primitives for Causal Interactions

Our architectural framework is based on the CCC architectural approach [2, 3], which includes a business micro-architecture to support engineers and system specifiers to model and implement evolving component-base systems. Architectural primitives like coordination contracts [2] model rules that determine how and when components need to interact in order to fulfil business requirements. The CCC can be classified as a coordination based approach [13] that borrows essential ideas from software architecture [19] in order to externalise interactions from computations, and superimposition as known from parallel program design [12] to support compositional evolution.

Indeed, in software architecture, modelling techniques have been proposed for supporting interaction-centric approaches. More precisely, such techniques promote interconnections to first-class citizens (architectural connectors) by separating the code that, in traditional approaches, is included in the components for handling the way they interact with the rest of the system, from the code that is responsible for the computations that are responsible for the services offered by the components.

The particular architectural approach that is adopted by the CCC builds on event-condition-action (ECA) rules for coordinating the joint behaviour that a group of components need to execute in reaction to a trigger generated by another component or outside the system. A so-called coordination law defines how a number of partners interact. The partners are not named: they are abstracted as coordination interfaces that define types of system entities in terms of operations that instance entities need to make available and events that need to be observed. As an example, consider the coordination of the way a doctor interacts with a respiratory-control system:

```
coordination interface respiratory-control
    partner type DEVICE
    types a:pressure, d:DOCTOR
    operations
        in-charge(d):Boolean
        verify():pressure
        decrease(a): post verify() = old verify()-a
        increase(a): post verify() = old verify()+a

coordination interface doctor-in-charge
    partner type DOCTOR
    types a: pressure
    events plus(a), minus(a)

coordination law restricted-respiratory
    partners d: doctor-in-charge, r: respiratory-control
    types a:pressure
    attributes  min,max:pressure
    rules
        when d.minus(a)
             with r.verify-a≥min and r.in-charge(d)
             do r.decrease(a)
        when d.plus(a)
             with r.verify+a•max and r.in-charge(d)
             do r.increase(a)
```

Each rule of the coordination law identifies, under the “when” clause, a trigger to which the instances of the law will react – e.g. a request by a doctor for an increase or decrease of the pressure. The trigger can be just an event observed directly over one

of the partners or a more complex condition built from one or more events. Under the "with" clause, we include conditions (guards) that should be observed for the reaction to be performed: that the changes in the pressure keep it within the specified bounds and that the doctor has been authorised to be in charge of the device. If any of the conditions fails, the reaction is not performed and the occurrence of the trigger fails. Explicit mechanisms can be defined for handling such failures.

There is neither a provision in the CCC approach, nor in any other architectural approach that we know, to model interactions that are only biddable, i.e. situations in which people (social components) are requested to perform given operations but the system cannot cause (force) them to perform these operations. For instance, biddable interaction would occur if the doctor would be requested to alter the current settings. In summary, one needs a richer model of interaction that can capture the fact that coordination in the presence of social components cannot be causal.

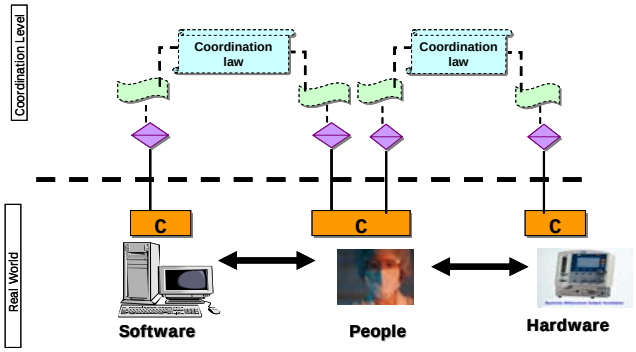

Figure 1: A CCC-based configuration

2.2 Architectural Primitives for Biddable Interactions

The social layer that is required for healthcare requires integrating a new collection of architectural primitives that support the modelling of human interactions in ways that are flexible and easily amenable to change. More precisely, for the kind of "just-in-time" binding and reconfigurations required for modelling biddability of human participants, we put forward another class of connector types (social laws) defined over a set of social roles, each of which represents the abilities of a social entity within the organisation.

2.2.1 Social Roles

Roles are abstract constructs that specify the behaviour expected of social components by means of operations and ascribed normative aspects that refer to certain institutionalised positions or capabilities. More concretely, we distinguish between having the ability to perform an operation and having the qualification or authorisation to do so: a social component may have the ability to perform an operation and still trigger a role violation if it is not an instance of a role that has the right qualifica-

tion. Here we use the word qualification to mean, for instance, that the organisation has empowered the social component to perform given operations.

As discussed below, the execution of operations by a component when playing a role without the required qualification is governed by a social law. A social law specifies (social) rules that either impose sanctions or provide a configuration in which the operation can be safely executed, depending on the context in which the violation takes place. However, we need to stress that the execution of operations, even if by qualified components, can be governed by coordination laws and, as such, be refused in certain circumstances for operational reasons, not for deontic ones.

We denote with [+] the operations for which the role is qualified. We can also define a subsumption relation between operations: by declaring $op_1 \supset op_2$ we mean that op_1 can only be executed as part of op_2, in which case a component qualified to do op_2 is also qualified to do op_1.

The general structure of a social role is as follows:

```
social role rolename {specialises rolename}
types {{par}+:datatype}*
operations {
    {'[+]'} opname {⊃ opname}
}*
```

For instance, GPs (General Practitioners) are qualified to perform routine tasks of seeing patients and registering for shifts in wards. A GP can also perform minor operations but will trigger a role violation unless he/she is an instance of a role that is qualified to do so.

We introduce the notion of *task* as an unordered set of events involved in the same activity. Among them, one or two events, corresponding to entry and exit operations, are considered as Behavioural Implicit Communications that need to be managed by the normative reconfiguration view (as captured by social laws) instead of causal coordination context. Our roles and coordination interfaces fit well together in the sense that they capture complementary aspects of human components: organisational and capability-related.

The overall importance of distinguishing between ability and qualification to perform an operation is that it reduces what are normally called normative positions [22] to deontically-governed role transitions. This is because, in the context of an architectural approach to system development, it is easy to model role transition in terms of dynamic reconfiguration. In the absence of such an explicit hierarchy, sub-ideal situations would have to be resolved just by means of sanctions. Instead, we take a more positive and active approach by enabling a reconfiguration if the current context allows such a deviation of the norm to be tolerated. This is precisely the goal of social laws as discussed below.

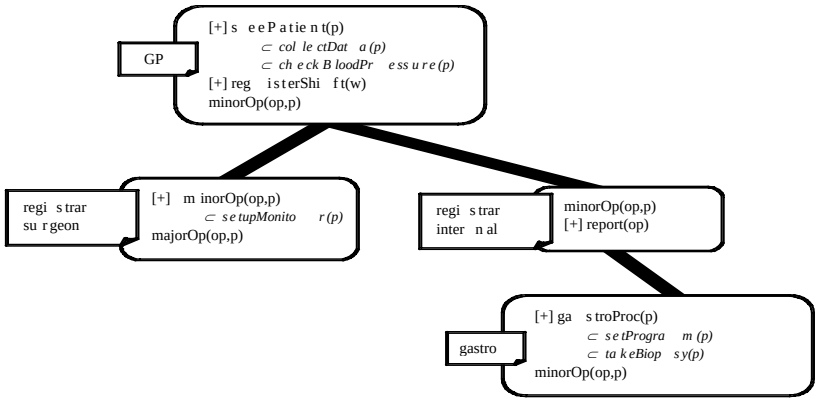

Figure 2: A role hierarchy

2.2.2 Social Laws

A social law is an architectural primitive through which a software architect models the circumstances in which a system is considered to be in a sub-ideal situation and how the system should respond to such a situation so as to preserve the overall integrity of the system. A sub-ideal situation occurs either as a result of a human deviating from a prescribed routine or by detecting a trigger that refers to a predetermined contextual change in the environment. A human may deviate from his prescribed behaviour either by acquiring a non-granted permission or by not obeying an imposed obligation that urges him/her to execute an action provided that appropriate architectural facilities are granted.

A social law captures interactions of a partner declared as an anchor role. Besides the anchor role, a social law identifies other partners through either social roles or coordination interfaces. The former are useful for reconfiguration operations and the latter for both detecting triggers and reconfigurations as explained below.

```
social law name
anchor role social role
partners {social role, coordination interface}*
types {{par}+:datatype}*
{violation rule
   when trigger
     if condition
     reconfiguration task
     sanction {operation}*
}*
```

In addition, a social law comprises one or more violation rules. We distinguish three kinds of triggers for violation rules: (1) operations of the anchor role that are executed by social components that have no qualification; (2) operations for which the anchor role is qualified but they are initiated in a context in which they are not permitted; (3) operations of the anchor role that are not executed in contexts in which they are required.

The first takes the form:

```
unqualified operation
```

The second takes the form:

```
operation and not enabling state
```

The third are of the form:

```
active state and not operation
```

Notice that, in order to detect the violation of the enabling state (permission), we need a coordination interface that provides an operation that returns a Boolean value and, in order to detect the violation of the obligation, we need a coordination interface that provides an event. The "negated operation" holds in the states in which the operation has not been scheduled for execution by the component that instantiates the anchor role.

2.2.3 A Motivating Example

This example was extracted from a survey undertaken at a Gastroenterology department[1] in UAE. By looking into their documentation we elicited a group of norms that affect the behaviour of doctors: (1) No operation can be undertaken without the patient's permission; (2) Surgical intervention should be carried by Surgeons[2] only; (3) In the case of emergency, a doctor may commit simple surgical interventions if surgeons are not available and it is a life-saving context.

As an example, consider the social law that applies to minor operations. Such procedures involve a social role – a GP – who is the anchor role in the sense that the social laws will apply to the actions performed by instances of this role, e.g. a gastroenterologist. In addition, three coordination interfaces are required to ensure that the GP interacts with the right components: the device that is monitoring the procedure – *monitor-procedure*, and the software component that provides access to administrative data – *administrator*. In the configuration of the system, there will be coordination laws modelling the way these components interact. Because of lack of space, we are not able to provide the definition of the relevant coordination interfaces and laws.

```
social law minor-operation
anchor role d:GP
type p:patient, op:operation
partners a:administrator, m:monitor-procedure
violation rule
    when d.minorOp(op,p) and not a.ensureConsent(op,d,p)
      if m.alarm(p)
      reconfiguration reconfMinor(d,op)
      sanction a.record(d,op,"no_consent")
    when unqualified d.minorOp(op,p)
      if m.alarm(p)
      reconfiguration reconfUnqual(d,op,p)
      sanction a.record(d,op,"unqualified")
```

The social law has two rules triggered by the same event: the moment in which a doctor initiates the operation on the patient. The first rule handles the situation in which there is no record of consent having been given by the patient for the doctor to perform that operation. If the monitor detects that there is an emergency situation,

[1] A specialized medical unit in Rashid Hospital, Dubai, U.A.E., that provides treatment for digestive diseases. The unit consists of an Endoscopies suite containing two modern fully equipped endoscopies rooms with ancillary supporting facilities for patient's reception, preparation and post-endoscopies recovery rooms.

[2] A reader should refer to Figure 2 for roles, tasks and permissions definitions.

then a reconfiguration of the context is performed to put in place the components and coordination contracts that are required for the operation to proceed. This may involve, for instance, providing access to further information registered on the patient's file, say on allergies. However, if the monitor does not detect an emergency, sanctions apply by recording the violation in the doctor's file.

The second rule is activated if the actual doctor is not qualified to perform minor operations, which is possible because the doctor's role matches one of the roles in the non-surgical branch of the doctor's role hierarchy: GP, registrar internal or gastro. In this case, we have to distinguish again if there is an emergency. For simplicity, we used the same alarm condition provided by the monitor. If an emergency is indeed detected, a reconfiguration of the context is performed to allow the doctor to proceed, for instance unblocking actions that, in normative states, should be forbidden to the doctor. Otherwise, sanctions apply. Notice that the reconfiguration operation takes the doctor as a parameter: the hospital may have different rules about the context that should be present during an operation depending on the type of doctor.

Notice that both rules can apply – the doctor may not be qualified and the patient may not have given consent. In the case of an emergency, both reconfigurations apply; otherwise, both sanctions are implemented. The subsequent section highlights reconfiguration operations and the formalism used to reason about them.

3 Modeling Reconfiguration Operations

In order to reason about system reconfigurations at a higher level of abstraction we require a representation of the system's configuration, its graph of components and connections at runtime, as well as of the rules governing their evolution. Graphs and graph transformation [15] provide a visually compelling yet mathematically rigorous formal technique that addresses our needs.

3.1 Modeling Configuration Graphs

Configuration graphs consist of nodes that refer to the system's social, mechanical and technological entities, and edges representing connectors that are superimposed on these entities to coordinate their interactions. More precisely, a configuration graph is a labelled graph where nodes are components labelled with instantiated interfaces and edges are connectors (contracts) labelled with the corresponding law type.

Valid system configurations (instances) are such that their graphs conform to constraints defined by type graphs – "filters" that restrict the allowed types of the instantiated nodes as well as types and cardinality of edges that connect them to populate an architecture instance.

We developed a specific graph typing structure to distinguish between configuration entities (nodes) whose corresponding permissions are fixed and some other entities that hold permissions amenable to change at runtime, e.g. social entities. This typing structure yields a twofold representation of the configuration graph that comprises a *components configuration graph* [20] and a *role configuration graph* that captures instances of roles, tasks and entry actions of social entities. More con-

cretely, the *components graph* is sufficient to reflect casual properties of software, mechanical components and their connections but it falls short of providing a suitable representation of human components that are biddable and subject to organisational norms that can be violated. Conversely, the *role graph* includes a biddable dimension and an organisational dimension; the former addresses the biddable nature of human components, which requires non-causal modelling primitives; the latter is constructed for modelling human capabilities and permissions within an organisation. The bridge between the two-configuration graphs consists of the common human nodes and the edge between the targeted task and its associated coordination interface copy that defines the signature of operations and services included in this task.

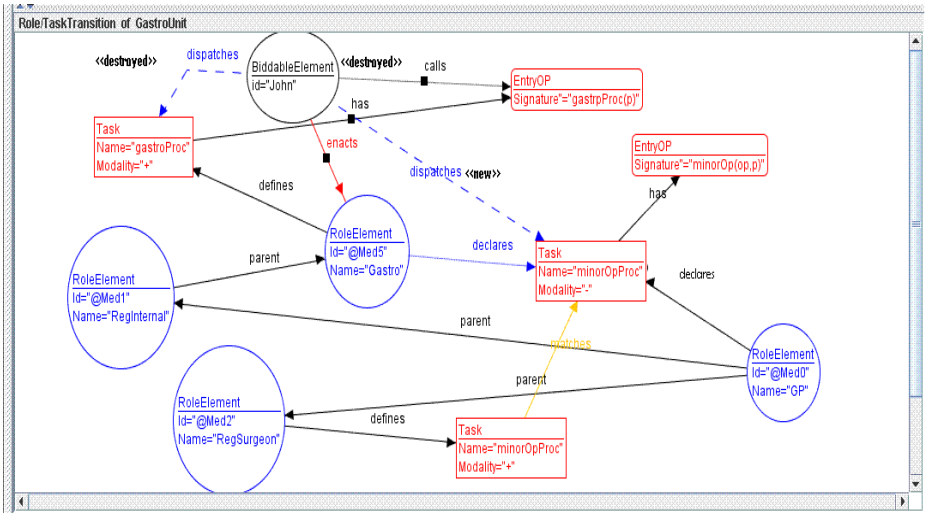

Figure 3: The configuration Graph

Figure 3 depicts a Configuration graph as a snapshot of the system configuration that preserves the constraints of the type graph: a biddable entity (John) calling an entry operation that is beyond his role's permissions.

3.2 Modeling Reconfiguration Rules Using Graph Transformations

Formally, a typed attributed graph transformation rule r: L→R consists of a pair of instance configuration graphs that conform to a type graph TG, and whose intersection is well-defined (this implies that edges that appear in both L and R are connected to the same vertices in both graphs, that vertices with the same name have to have the same type, and so on). The left-hand side L represents the pre-conditions of the rule while the right-hand side R describes the post-conditions. The AGG[3] tool has been used to demonstrate configurations and reconfigurations as graphs and transformation rules.

[3] The Attributed Graph Grammar System: http://tfs.cs.tu-berlin.de/agg/

We devised two types of reconfigurations: *local reconfigurations* operations that target human (biddable) components by writing his/her permissions or obligations; *global reconfigurations* make a full transition to a new workflow entry (an initial task). For example, *reconfUnqual(role/subject, operation, target)* performs a reconfiguration operation on a human role/task space and its bound permissions, whereas *reconfGloabl(role, Process, entry_operation)* manipulate both software and hardware entities configuration to allow the human participant to perform a new set of operations to achieve new goals as a response of new context settings.

Both reconfiguration operations provide operational semantics for social laws by allowing humans, participating in an instance configuration, to deviate from the prescribed process pattern, e.g. *medical pathways*, at the execution time and/or performing a *g*lobal reconfiguration: simply model the transition from the current configuration (process) to another.

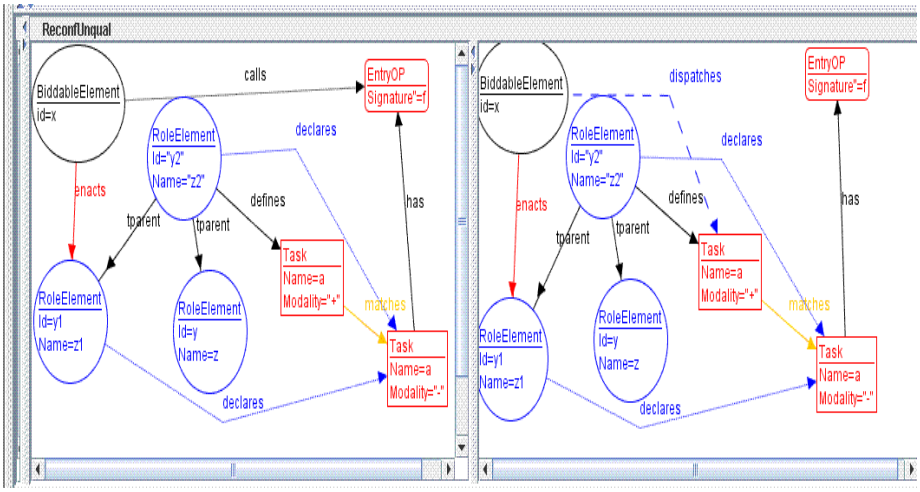

Figure 4: *ReconfUnequal* Transformation rule

For simplicity and due to space limitations we provide a simple transformation rule (Figure 4) that illustrates a generic and role-based reconfiguration operation – *reconfUnqual* – in which a GP is allowed to perform a minor surgery – *tracheostomy* – in case of emergency, provided that the preconditions are tested by the corresponding social law body when the entry operation is fired. Labels <<*destroyed*>> and <<*new*>> in Figure 3 designate deleted and created nodes/edges imposed by the corresponding transformation rule.

The formalisation provided through graph transformations supports reasoning about reconfiguration steps, brings together computation and coordination with process management in a single model and facilitates the derivation of overall system behaviour by exploiting the notion on operational semantics c.f. GOS [9].

4 Concluding Remarks

The main contribution of our research has been to leverage modelling primitives developed for software architectures to cater for interactions that involve human components. Our approach takes into account the biddable, non-causal nature of human actions, and provides a mechanism for adjusting permissions and obligations on interactions between technical and human components so as to react and adapt to changes in the environment in which they operate. Similarly, adaptation may result from monitoring services [4] and imply a reconfiguration of roles and interactions. Our approach follows a normative system perspective and includes: social laws that express reactions to non-normative situations that may arise from violation of permissions or obligations; social roles that capture humans capabilities within organisations; operations that, within social laws, can reconfigure interactions and/or reassign roles to human components.

In the future, we would like to explore the synergies between our architectural approach and recent research on workflow-based systems, which have been shifting from Workflow Control Patterns and Workflow Data Patterns to that Workflow Resource Pattern [20]. The latter come in line with our focus on modelling resources (human/non human) and their interactions. Recent research on workflow has also shed light on the need for runtime changes – *momentary changes* in [23]. Additionally, Lenz et al. [17] stated that there is a price to be paid for isolating control flow from application logic. They put forward a workflow engine to accommodate ad-hoc changes at different levels of abstractions. Although they distinguish between stable organizational processes and continuously changing clinical treatment processes, they are not able to model how knowledge about medical staff may affect such changes.

References

1. Allen, R. and D. Garlan, "A Formal Basis for Architectural Connectors", *ACM TOSEM*, vol. 6, no. 3, pp. 213-249, 1997.
2. Andrade, L. F., and J. L. Fiadeiro, Architecture Based Evolution of Software Systems, in M. Bernardo and P. Inverardi (eds.), *Formal Methods for Software Architectures*, LNCS 2804, Springer-Verlag, pp. 148-181, 2003.
3. Andrade, L. F., J. L. Fiadeiro and M. Wermelinger, Enforcing Business Policies through Automated Reconfiguration, in *Proc. of the 16th Intl. Conf. on Automated Software Engineering*, IEEE Computer Society Press, pp. 426-429, 2001.
4. Baresi, L., C. Ghezzi and S. Guina, Smart Monitoring for Composed Service, in *Proc. of the 2nd International Conference on Service Oriented Computing*, pp. 15-19, 2004.
5. Brier, B., L. Rapanotti and J. Hall, Problem Frame for Socio-Technical Systems: predictability and change, in *Proc.* of *1st International Workshop on Advances and Applications of Problem Frames (WAAPF 2004)*, *ICSE*, Edinburgh, Scotland, 2004.
6. Castelfranchi, C., and F. Giardini, Silent Agents. Behavioural Implicit Communication for M-A Coordination and HMI, in *Proc. of the 2nd Annual Symposium on Autonomous Intelligent Networks and Systems*, Menlo Park, CA, 2003.

7. Colman, A. W. and J. Han, Organisational Roles and Players, in *Proc. of AAAI'05 Fall Symposium on Roles*, An Interdisciplinary Perspective, Arlington, VA, pp. 55-62, 2005.
8. El-Hassan, O. and J. L. Fiadeiro, Role-based Architectural Modelling of Socio-Technical Systems, In *Proc. of the 2nd International Workshop on Coordination and Organisation (CoOrg'06)*, ENTCS, vol. 181, pp. 5-17, 2007.
9. Engels, G., J. H. Hausmann, R. Heckel, and S. Sauer, "Dynamic Meta Modelling: A Graphical Approach to the Operational Semantics of Behavioural Diagrams in UML", in A. Evans, S. Kent and B. Selic (eds.), *UML 2000*, LNCS, Springer, vol. 1939, pp. 323–337, York, UK, 2000.
10. Falcone, R., and C. Castelfranchi, Level of Delegation and Levels of Adoption as the basis for Adjustable Autonomy, in *Proc. of the 6th AI*IA*, LNCS, vol. 1792, pp. 273-284, 1999.
11. Fiadeiro, J. L., "Designing for Software's Social Complexity", *Computer*, vol. 40, no. 1, pp. 34-39, 2007.
12. Fiadeiro, J. L. and T. Maibaum, "Categorical Semantics of Parallel Program Design", *Science of Computer Programming*, vol. 28, pp. 111-138, 1997.
13. Gelernter, D. and N. Carriero, "Coordination Languages and their Significance", *CACM*, vol. 35, no. 2, pp. 97-107, 1992.
14. Grudin, J., "CSCW – History and Focus", *IEEE Computer*, vol. 27, no. 5, pp. 19-26, 1994.
15. Heckel, R., Graph Transformation in a Nutshell, in *Proc. of the School of SegraVis Research Training Network on Foundations of Visual Modelling Techniques (FoVMT 2004)*, LNCS, vol. 148, no. 1, pp. 187-198, 2006.
16. Jackson, M., Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices, Addison Wesley, 1995.
17. Lenz, R., and M. Reichert, "IT Support for Healthcare processes: premises, challenges, perspectives", *Data & Knowledge Engineering*, vol. 61, no. 1, pp. 39-58, April 2007.
18. Padmanabhan, V., G. Governatori, S. Sadiq, R. Colomb, and A. Rotolo, Process Modelling: The Deontic way, in M. Stumptner, S. Hartmann and Y. Kiyoki (eds.), *the 3rd Asia Pacific Conference on Conceptual Modelling (APCCM 2006)*, 2005.
19. Perry, D. E., and A. L. Wolf, "Foundation for the Study of Software Architectures", *ACM SIGSOFT Soft. Engineering Notes*, vol.17, no. 4, pp. 40-52, 1992.
20. Russel, N., W. M. P. van der Aalst, A. H. M. ter Hofstede and D. Edmond, Workflow Resource Patterns Identification, Representation and Tool Support., in *Proc. of CAiSE'05*, LNCS, Springer, vol. 3520, pp. 216-232, 2005.
21. Sandhu, R. S., E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models", *IEEE Computer*, vol. 29, pp. 38-48, 1996.
22. Sergot, M., "Normative Positions", in H. Prakken and P. McNamara (eds.), *Norms, Logics and Information systems: new studies in Deontic Logic*, IOS Press, pp. 289-310, 1998.
23. van der Aalst, W. M. P., and D. Jablonski, "Dealing with Workflow Change: identification of issues and solutions", *Comput Syst Sci & Eng*, vol. 15, no. 5, pp. 267-276, 2000.
24. Wermelinger, M., A. Lopes and J. L. Fiadeiro, A Graph based Architectural Reconfiguration Language, in *Software Engineering- ESEC/FSE'01*, ACM Press, pp. 21-32, 2001.

Adaptive Workflows for Healthcare Information Systems

Kees van Hee, Helen Schonenberg, Alexander Serebrenik, Natalia Sidorova, and Jan Martijn van der Werf

Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{k.m.v.hee, m.h.schonenberg, a.serebrenik, n.sidorova, j.m.e.m.v.d.werf}@tue.nl

Abstract. Current challenges in Healthcare Information Systems (HIS) include supplying patients with personalized medical information, creating means for efficient information flow between different healthcare providers in order to lower risks of medical errors and increase the quality of care. To address these challenges, the information about patient-related processes, such as currently executed medical protocols, should be made available for medical staff and patients. Existing HIS are mostly data-centered, and therefore cannot provide an adequate solution. To give processes a prominent role in HIS, we apply the adaptive workflow nets framework. This framework allows both healthcare providers and patients to get an insight into the past and current processes, but also foresee possible future developments. It also ensures quality and timing of data communication essential for efficient information flow.
Keywords: adaptive workflows, EPR, medical protocols, Healthcare Information Systems.

1 Introduction

The recent study of the Netherlands Health Care Inspectorate [23] has established a number of serious shortcomings in the communication between healthcare providers that can cause risks for patient safety. The typical examples named in the study are a lack of communication between the anesthesiologist and the surgeon involved in the same surgery; repetitive overwritings of variable data, such as blood pressure, with no information when, in which circumstances and by whom the measurements were performed; the use of “no message — good message” principle in the communication between specialists; providing insufficient information for patients about their treatments and expected developments. Another tendency reported is the steady increase of the total amount of patient-related information available, which, on one hand, increases the survival rate, but on the other hand, complicates the overall picture due to the chaotic nature of the information. The chance that the surgeon is aware of all information relevant to the surgery being performed is estimated as very low. The problems mentioned indicate a great need in a new generation of Health Information Systems (HIS) that would satisfy new information and communication demands.

At the same time, the patient should become a focal point of the new generation HIS. On April 30 of 2004, the European Commission adopted an action plan [7] aiming at making healthcare better for European citizens. As opposed to currently available provider-centered health systems, the action plan envisions citizen-centered health systems. A survey of Harris Interactive and ARiA Marketing conducted in 2000 [15] shows that more than 80% of the respondents are interested in obtaining on-line personalized medical information and electronic alerts specific to their medical histories, while 69% would like to have access to the charts that monitor the progress. Therefore, in this paper we aim at the creating a new concept of HIS that would increase the availability of personalized information in HIS both for care providers and for patients.

Currently employed HIS concentrate often on patient-related *data* rather than consider treatments as *processes* producing these data. By getting access to the process information, a patient can obtain a clear personalized picture of ongoing treatments, expectations and risks. Therefore, (s)he is more likely to become an informed decision-maker [8]—note that "participatory" decision-making model is recommended as the preferred model of treatment decision-making [4]. Moreover, shifting the focus of healthcare information systems from data to processes provides a practitioner with information on continuation of a treatment initiated by her and continued elsewhere. That is another reason why we advocate the move from data to processes.

To address the issues raised above, we propose a new approach to HIS. The key idea consists in associating a (number of) process(es) to each patient. By logging into the private web area, the patient gets access to the information related to processes associated to her. This information should include (a view on) data produced by previously performed treatments and a number of likely scenarios for the treatment continuation. Using the same information, a care provider can decide to alter an ongoing process, to abort such a process or to initiate a new one. New processes can either be suggested by a physician, or, more probably, they can be borrowed from a library of protocols, including e.g., hospital policies and medical guidelines.

Our approach relies on the framework of adaptive workflow nets [10, 12]. Main advantages of the framework include adaptivity, adaptability and separation of concerns. By *adaptivity* we understand the ability of a process to modify itself as opposed to *adaptability*, which is the ability of a process to be modified by an external party. By *separation of concerns* we understand that every (sub)process has its owner, and the owner is the only actor responsible for performing the steps of the process. Still, the processes are interrelated and communicating.

In addition, we extend the currently used concept of the electronic patient record (EPR) with the notion of history. In other words, rather than overwriting the blood pressure in a database cell, we record the measurement every time it has been taken, by whom (the owner of the process), when and why (note that this information can be automatically generated since we know to which process this action belongs). This history is then used in the decision making process. It should be noted that the availability of history also enables the application of data and process mining techniques [2, 17], which can lead to improving the quality of medical care.

2 Motivating example

As a motivating example we consider the story of Saskia, a thirty-six years old Dutch preschool teacher, pregnant with her first child in the fifth week gestation. Saskia occasionally uses her home computer for chatting with her friends or looking up information on the Internet. During her pregnancy Saskia is assisted by a number of healthcare professionals: midwives, lab assistants, doctors, having (partial) access to Saskia's (electronic) patient record. Below we present the first steps of Saskia's pregnancy in the current situation and discuss possible improvements due to the implementation of our approach.

Currently Saskia looks up the information on midwifes' offices in an on-line telephone directory. After consulting the web-sites of the offices, she selects one, and calls the office to arrange an appointment with a midwife. During the first visit, the midwife records Saskia's medical, social and gynaecological history and orders a number of standard lab tests: blood type, rhesus, iron deficiency, glucose level and urine. Moreover, since Saskia is thirty-six years old, the midwife briefly informs her on possible age-related risks for the baby and additional tests that can be performed. Upon receiving this information Saskia gets overanxious: she has no time to reflect on the matter and she is not able to decide whether she is willing to take these tests.

Getting home Saskia talks to her husband and looks for additional information available on-line. Based on this information she decides to undergo a nuchal translucency scan (NT screening). Saskia calls the midwifes' office to arrange another appointment at which she would order the test.

Blood tests show that Saskia is Rh(D) negative, while her husband is Rh(D) positive. Therefore, at twenty eight weeks gestation she gets an anti Rh(D) IgG immunoglobulin injection, which will be repeated after the delivery.

Desired After selecting the office, Saskia fills an on-line form with her personal data. Being prompted what means of communication (e-mail, text messages, regular mail, phone call) does she prefer, Saskia chooses text messages. Indeed, Saskia is very excited about her pregnancy and wants to get all the information as soon as possible whether she is at home, at her office or out with friends. However, she is a working woman and not every moment might be appropriate for getting the information. Figure 1 presents the completed registration form.

Saskia receives a text message informing her which standard tests she should undergo. She also gets a link to a *personalized web-page* providing her with information on age-related risks and additional tests. Saskia takes her time to study the web-page together with her husband and decides to undergo an NT screening. She indicates this choice on her personalized web-page. A number of test times is proposed to her taking into account the scanner availability and that NT screening is performed between the eleventh and the fourteenth weeks gestation. The time selected and additional information are sent to the laboratory working with the midwifes' office.

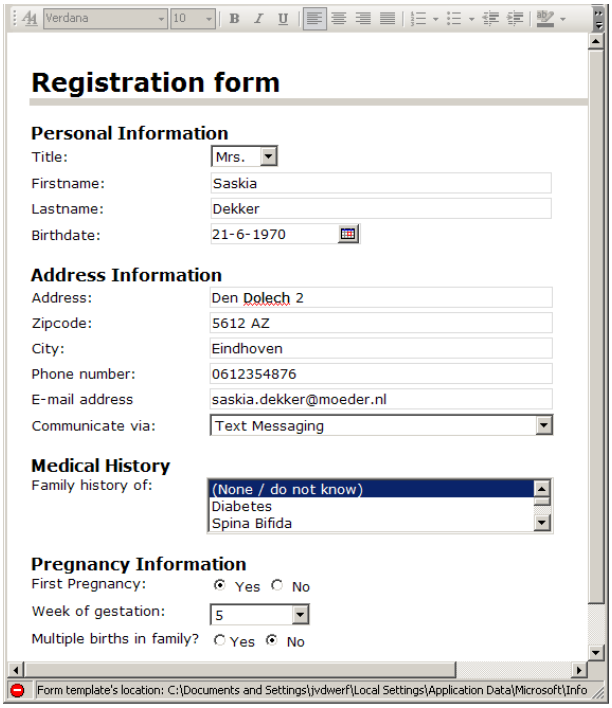

Fig. 1. Registration on-line form

Moreover, Saskia gets informed on possible future developments. For instance, she learns that if, according to the NT screening results, a chance of her baby being affected is high, she will be offered to undergo amniocentesis to obtain a conclusive evidence.

The personalized web-page further includes a personalized pregnancy calender. All appointments arranged are automatically added and reminders are sent. Furthermore, depending on Saskia's personal data and outcomes of the preceding tests, new appointments to be scheduled are highlighted in the corresponding periods. For instance, anti Rh(D) IgG immunoglobulin injections will appear on the twenty eighth week and in her delivery procedure description. Note that healthcare providers have access to Saskia's pregnancy calender and get alerted, for example, if she fails to show up at her anti Rh(D) IgG immunoglobulin injection date.

Requirements imposed by the example In order to support the desired scenario presented above a number of changes has to be introduced to current healthcare information systems. First of all, in order to provide Saskia with information on possible future developments, a healthcare information system should be made aware of the ongoing *processes* rather than only *data* involved. Recall that data relates to information till the

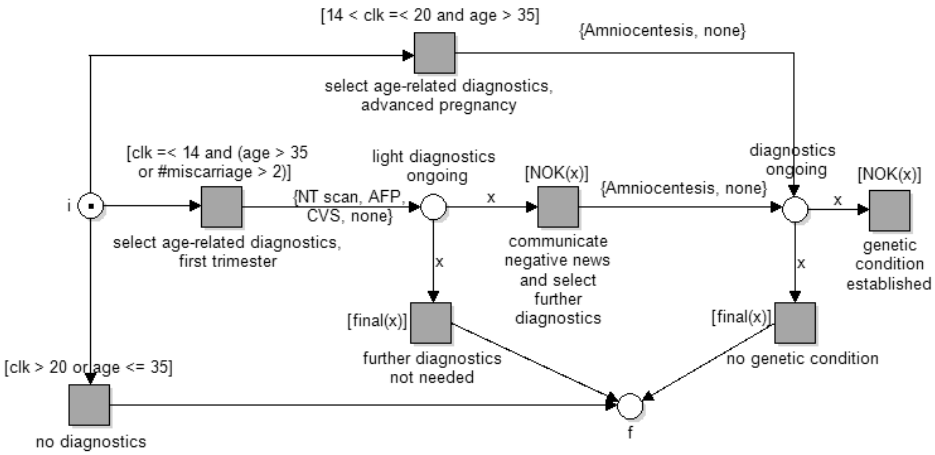

Fig. 2. Genetic condition related tests

current moment, while patients can be eager to know what can happen next. However, assuming one ongoing process is not realistic, as different healthcare providers have their own rules and processes. Sharing these processes might be superficial (a midwife should not necessarily be knowledgeable of the biochemistry of a blood test) or undesirable (lab assistant should not have access to Saskia's personal information). Therefore, rather than considering one process we envision *a series of interrelated but independent processes*.

Another important conclusion that we can draw from our example is that processes should be able to modify themselves on-the-fly. For instance, when Saskia's Rh(D) turns out to be negative, special treatment should be performed, i.e., a special process should be initiated in parallel with the ongoing one. We refer to the ability of a process to modify itself on-the-fly as *adaptivity*. Still, we do not intend to substitute medical stuff by an automatic decision making system. The choice of treatment protocols is certainly made by the care providers together with the patient. Therefore, processes should be not only adaptive but also *adaptable*.

3 Adaptive Petri Nets

As we could see in the Saskia example, HIS is a domain with a great need for adaptivity. In this section we present the theoretical foundations of our approach, so called *adaptive Petri nets*, and we illustrate the concepts introduced by means of a running example: a simplified version of the Dutch age-related prenatal diagnostics protocol, given in Figure 2. The owner of the considered process is the midwife.

A Petri net [18] is a bipartite graph whose nodes are called *places* and *transitions*. Transitions, graphically represented as rectangles, correspond to actions being taken.

Places are represented as circles, and they are used to define the process flow. Given a transition we distinguish the *input places of the transition*, i.e., places with an arc going to the transition, and *output places of the transition*, i.e., places with an incoming arc coming from the transition. The Petri net presented in Figure 2 includes such transitions as "select age-related diagnostics, first trimester" and "communicate negative news and select further diagnostics". The name of a transition is written *under* the corresponding rectangle.

Workflow nets [1] are a special class of Petri nets, well-suited for modelling workflow processes. Workflow nets have exactly one place with no incoming arcs, called *the initial place*, and exactly one place with no outgoing arcs, called *the final place*. Moreover, every node in a workflow net is on a path from the initial place to the final place. *Extended workflow nets* [10] can be obtained from a workflow net by adding *exception transitions*, which are transitions with at least one input place and no output places. Exception transitions are used for modelling undesirable, abnormal or irregular events of such a nature that the process cannot decide itself how to continue and an assistance of a higher authority/layer is required. The Petri net in Figure 2 is an extended workflow net with the initial place *i*, the final place f and the exception transition "genetic condition established".

The state of a system is represented by means of *tokens*, drawn as black dots and residing in places. The initial state consists of a single token in the initial place *i*, while the final state consists of a single token in the final place f. The final state corresponds to the most likely, expected, termination of the process. Dynamics of a process is expressed by means of the *token game*: performing an action corresponds to a *transition firing* removing a token from every input place and adding a token to every output place of the transition. For instance, "select age-related diagnostics, first trimester" removes one token from the initial place *i* and produces one token in the output place *light diagnostics ongoing*. Figure 2 shows the state of the system before a firing of "select age-related diagnostics, first trimester". In the normal course of events, process terminates when we reach the state consisting of a single token on place f. Firings of exception transitions terminate the execution of the process independently of the process state, disregarding the fact whether there are still tokens left and transitions enabled.

Colored Petri nets [16] extend Petri nets by introducing data and time into the model, i.e., allowing to model a data flow in addition to a control flow. Due to historical reasons data types are commonly referred to as *colors*. Classical Petri nets are extended there by *guards* and *arc inscriptions*. The guard is a logical expression determining whether the transition may fire. Arc expressions at the outgoing arcs define data transformations. The firing of transitions become thus data dependent, and, moreover, transitions modify data.

Global history nets [12] further extend Petri nets by assuming the availability of a history record, registering all firings and the time of firing together with the information which process performed it. Transition guards can depend on the information contained in the history record. Guards are written between square brackets. For instance, "select age-related diagnostics, first trimester" has the following guard: "$\mathsf{clk} \leq 14$ *and* $(\mathsf{age} > 35$ *or* $\#\mathsf{miscarriage} > 2)$". This means that the corresponding EPR contains information that the gestation week is not later than 14 (clk denotes here the pregnancy clock), and

either the maternal age exceeds 35 or the history record contains at least three previous miscarriages.

Adaptive nets[10] further extend the formalism by introducing a special color: nets. In other words, a token can be associated with another (adaptive) net, called a *token net*. Token nets are created by using suggested *library nets* or tailor-made nets. So, the firing of "select age-related diagnostics, first trimester" is in fact a procedure where the midwife together with the pregnant woman decides which prenatal test procedure will be carried out, if any at all. The suggested tests (NT scan, AFP, CVS and none) are indicated on the outgoing arc of the transition. By using "any" in the suggested set of token nets, we allow to take an arbitrary, possibly tailor-made, protocol. The considered example does not make use of these freedom of choice, since the set of tests is predefined. Note that the owner of the created token net is not necessarily the midwife. In this example, it will be a lab performing the selected test procedure.

Guards can be used to *synchronize* firings of the (upper layer) net with firings of a token net. The transition "communicate negative news and select further diagnostics" fires if the selected test procedure reports a high risk of genetic condition (NOK), for it is conditioned by the guard $\mathrm{NOK}(x)$, where *x* refers to the token consumed from the "light diagnostics ongoing" place. In other words, NOK in the selected test procedure and "communicate negative news and select further diagnostics" fire *synchronously*, which corresponds to the communication between the lab and the midwife. An additional form of synchronization is illustrated by "further diagnostics not needed". This transition can fire if the token net in the "light diagnostics ongoing" place has reached its final state, i.e., the only token present in the token net is residing in its final place.

4 Modelling Care Processes with Adaptive Nets

In this section we discuss how adaptive nets, discribed in Section 3 can be used to model care processes discussed in Section 2. First of all, we assume that (a view on) the electronic patient record (EPR) is available for all nets. EPR can be seen as a standard data record extended with the history log. Access to the EPR can and typically is further restricted depending on the task being performed: all tasks should be able to consult the pregnancy clock, but it is set during the registration and can be corrected by the midwife only.

Although the theoretical framework of adaptive nets does not restrict in the depth of nesting, we envision that the care applications will typically make use of three-layered processes. The *top-level* process represents the main process flow: registration, the choice of diagnostics and treatment protocols and closing the case. The main process flow can be associated with strategic goals. The second layer is normally a protocol layer. Processes of the second layer can can be carried out by the main care provider herself or delegated to other care providers. The operations can be seen as implementing the tactic goals of the process. The net at Figure 2 provides an example of a second level net. Finally, realization of medical protocols can require services of parties such as medical labs. These basic steps constitute the *third and the last level* of the process.

The deviations from the described process architecture scheme are of course possible: the midwife could for example demand Saskia to consult a cardiologist in case her

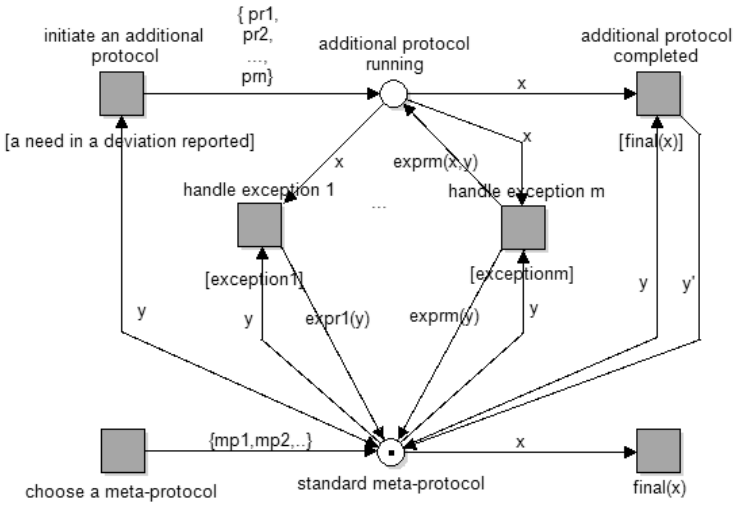

Fig. 3. Generic schema of a subprocess.

blood pressure is repeatedly evaluated as too high. The cardiologist process, located at the second layer in this case, is however a strategic-level process, which is still a service for the midwife protocol. The nesting depth in this case could be greater than three.

The top-level process consists typically of the initialization (registration, analysis of the history available), a number of subprocesses taking care of adaptivity aspects, and a process termination (including a semi-automatic post-processing of the EPR to provide consistent data for future use). Figure 3 shows a pattern for a subprocess. The transition “initiate an additional protocol” is mostly often triggered by some event in the environment (e.g. an additional complaint of a patient). In response to this event, an additional protocol can be chosen and started (alternatively, or additionally, the token net on the place “standard meta-protocol” can be modified). Exceptions and the termination of this additional protocol are then processed by the corresponding transitions by initiating new processes of modifying the running protocol. Note that there are two kinds of exception-guards possible — the first one are guards demanding a synchronization with a corresponding exception transition in the token net, while the second one are guards specifying some external trigger.

The history record is composed from all actions taken in each protocol and it provides not only the information on the action taken but also specifies who (which process) took it. Also the initiations of additional protocols and modifications in the running protocols are logged. This allows to keep the (otherwise chaotic) history record well-structured, since we can always make a query to get all the information related to some treatment, abstract from unnecessary details (e.g. do not show actions of the third layer protocols), aggregate information related to a class of treatments (e.g. cardiolog-

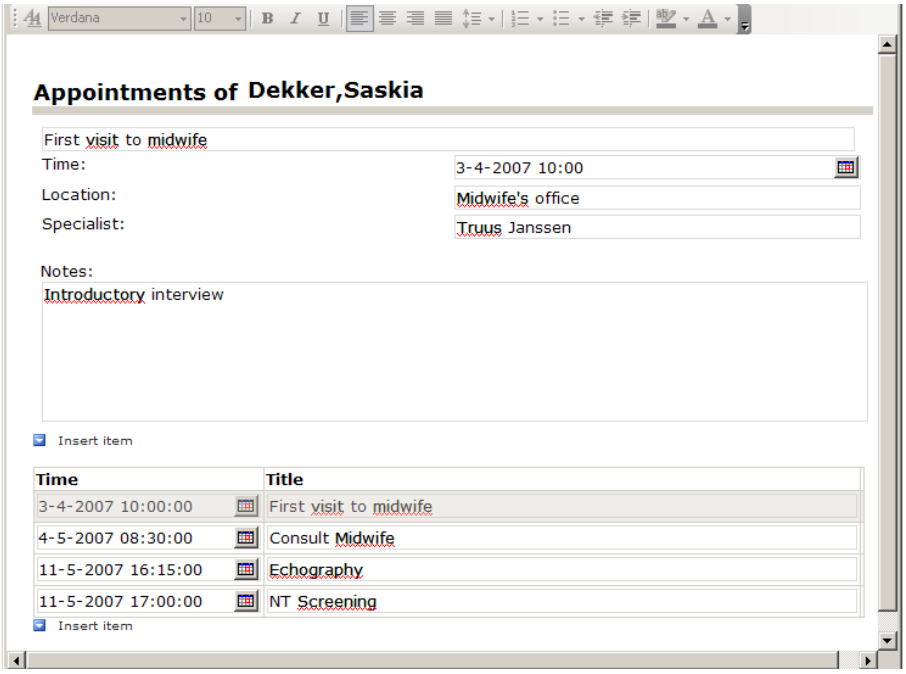

Fig. 4. Saskia's appointments calendar

ical treatments), even if these treatments were initiated by different care providers (the midwife, the GP, the cardiologist).

Having the information on the processes running, we can construct a forecast view for both care providers and the patient by constructing and exploring a (partial) state space of the running protocols. Here different options are possible. The most simple one is constructing a forecast under the assumption that no exception will happen (e.g. the normal course of pregnancy, taking into account personal ERP information, like negative Rh(D)). More elaborated views allows to look into the future taking into account exceptions that are predefined in the running protocols. For instance, when the age-related diagnostics protocol in Fig. 2 is started for Saskia, it is possible to build a view informing that amniocentesis can be performed in case the NT-screening reports a high risk of a genetic condition.

The first type of a forecast view (the likely one) can also be used for such a pragmatic thing as planning appointments, tests and treatments. An important issue here is that the interprocess dependencies and restrictions can be handled, so that different protocols would not interfere or damage each other. Another trivial gain is e.g. providing a personalized pregnancy calendar for Saskia (see Figure 4) and providing her with reminders on appointments and tests supplied with additional information, like prerequisites of these tests.

5 Conclusion

The major contribution of our work consists in proposing a process-centered patient-centered framework for the new generation of HIS. While patient-centered systems attract more and more attention of the research community, the current proposals [5, 21] concentrate mostly on implementation issues, such as data communication and heterogeneity of the application platforms, rather than on the conceptual ones, such as separation of concerns and adaptivity.

Our proposal is based on a solid theoretical foundation, namely, adaptive (nested) nets [11, 10], a subclass of Petri nets. This allows us to perform a number of automatic correctness checks, like soundness and circumspectness. Soundness means that every process can terminate properly from any reachable state, while circumspectness means that every exception can be taken care of by the higher layer.

We do not expect the end user to know what Petri nets are and aim at providing a user-friendly web-interface instead. For this purpose, we developed the tool YasperWe [9] that is designed for prototyping IS. The tool integrates Yasper, which is a Petri net editor including a number of analysis options and compatible with some other analysis tools, with Microsoft Infopath.

Related work Processes in healthcare are commonly documented in the form of *medical guidelines*. A guideline is not limited to doctors but also covers the workspace of nurses and paramedical personnel. There have been several attempts to formalize guidelines as *flowcharts* and *decision diagrams* and incorporate them into medical decision support systems.

Petri nets have been used for modeling of healthcare workflow, also known as careflow [13, 19, 20]. The guideline execution system GUIDE [20] translates formalized guidelines to a hierarchical timed colored Petri net. The resulting net can be run to simulate the implementation of the guideline in clinical setting. However, this formalism misses adaptivity and separation of concerns. The idea of adaptivity, i.e., controlled modification, in Petri nets has been considered in [6, 14]. However, these approaches were able to model only processes involving two care providers, for instance, a midwife and a general practitioner, which is not sufficient for common healthcare processes.

Currently, there are several approaches that offer some degree of flexibility. Many share the idea of modeling with underspecification, i.e. a model where parts of the process are not given explicitly, but represented by a placeholder. At run-time the configuration is completed by binding the placeholder to a process from a repository, which is also known as late binding. Adaptive nets [10, 11], worklets [3], pockets of flexibility [22] are modeling techniques that allow for late binding. Three main advantages of adaptive nets compared to the other approaches are the following:

- the ability to modify the structure of running processes in a controlled fashion;
- the possibility to define explicit synchronization between a token net and its owner;
- verification of a number of correctness properties for a subclass of adaptive nets.

Future work For the future work, we are going to propose a number of process patterns to facilitate process modelling and a number of query patterns for the creation of different user views.

References

1. W. M. P. van der Aalst. Verification of workflow nets. In P. Azéma and G. Balbo, editors, *Application and Theory of Petri Nets 1997, ICATPN'1997*, volume 1248 of *Lecture Notes in Computer Science*. Springer, 1997.
2. W. M. P. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004.
3. M. Adams, A. H. M. ter Hofstede, D. Edmond, and W. M. P. van der Aalst. Worklets: A service-oriented implementation of dynamic flexibility in workflows. In R. Meersman and Z. Tari, editors, *OTM Conferences (1)*, volume 4275 of *Lecture Notes in Computer Science*, pages 291–308. Springer, 2006.
4. C. Charles, A. Gafni, and T. Whelan. Shared decision-making in the medical encounter: What does it mean? (or it takes at least two to tango). *Social Science and Medicine*, 44(5):681–692, Mar. 1997.
5. P. Cheng, C. Yang, H. Chen, S. Chen, and J. Lai. Application of hl7 in a collaborative healthcare information system. In *Engineering in Medicine and Biology Society, 2004. EMBC 2004. Conference Proceedings. 26th Annual International Conference of the*, volume 2, pages 3354–3357, Sept. 2004.
6. H. Ehrig and J. Padberg. Graph grammars and Petri net transformations. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 496–536. Springer, 2003.
7. European Comission. Communication from the commission to the council, the european parliament, the european economic and social committee and the committee of the regions. e-health - making healthcare better for european citizens: An action plan for a european e-health area, Apr. 2004. SEC(2004)539.
8. B. S. Gerber and A. R. Eiser. The patient-physician relationship in the internet age: Future prospects and the research agenda. *Journal of Medical Internet Research*, 3(2):e15, 2001.
9. K. van Hee, J. Keiren, R. Post, N. Sidorova, and J. M. van der Werf. Designing case handling systems. In *Proc. of the International Workshop on Petri Nets and Software Engineering*, 2007.
10. K. van Hee, I. Lomazova, O. Oanea, A. Serebrenik, N. Sidorova, and M. Voorhoeve. Nested Nets for Adaptive Systems. In *Proc. of the 27th International Conference on Application and Theory of Petri Nets*, volume 4024 of *LNCS*, pages 241–260, 2006.
11. K. M. van Hee, I. A. Lomazova, O. Oanea, A. Serebrenik, N. Sidorova, and M. Voorhoeve. Checking properties of adaptive workflow nets. *Fundamenta Informaticae*. accepted.
12. K. M. van Hee, A. Serebrenik, N. Sidorova, and W. van der Aalst. History-dependent Petri nets. In *Petri Nets and Other Models of Concurrency - ICATPN 2007, 28th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency*, volume 4546 of *Lecture Notes in Computer Science*, pages 164–183. Springer, 2007.
13. K. Hoffman. Run time modification of algebraic high level nets and algebraic higher order nets using folding and unfolding construction. In G. Hommel, editor, *Proceedings of the 3rd Internation Workshop Communication Based Systems*, pages 55–72. Kluwer Academic Publishers, 2000.
14. K. Hoffmann, H. Ehrig, and T. Mossakowski. High-level nets with nets and rules as tokens. In G. Ciardo and P. Darondeau, editors, *ICATPN*, volume 3536 of *Lecture Notes in Computer Science*, pages 268–288. Springer, 2005.

15. Q. Homan. Harris interactive and ARiA marketing healthcare satisfaction study, Oct. 2000. Available at http://www.harrisinteractive.com/news/downloads/HarrisAriaHCSatRpt.PDF.
16. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*. Monographs in Theoretical Computer Science. Springer-Verlag, 1997.
17. H. C. Koh and G. Tan. Data mining applications in healthcare. *Journal of Healthcare Information Management*, 19(2):64–72, 2005.
18. C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, University of Bonn, Bonn, West Germany, 1962.
19. S. Quaglini, S. Panzarasa, A. Cavallini, G. Micieli, C. Pernice, and M. Stefanelli. Smooth integration of decision support into an existing electronic patient record. In S. Miksch, J. Hunter, and E. T. Keravnou, editors, *AIME*, volume 3581 of *Lecture Notes in Computer Science*, pages 89–93. Springer, 2005.
20. S. Quaglini, M. Stefanelli, A. Cavallini, G. Micieli, C. Fassino, and C. Mossa. Guideline-based careflow systems. *Artificial Intelligence in Medicine*, 20(1):5–22, 2000.
21. F. L. Ricci and L. D. Serbanati. Mobidis: Toward a patient centric healthcare information system. *Studies in Health Technology and Informatics*, 116:557–562, 2005.
22. S. W. Sadiq, W. Sadiq, and M. E. Orlowska. Pockets of flexibility in workflow specification. In *ER '01: Proceedings of the 20th International Conference on Conceptual Modeling*, pages 513–526, London, UK, 2001. Springer-Verlag.
23. G. van der Wal. Rapport preoperatief traject ontbeert multidisciplinaire en gestandaardiseerde aanpak en teamvorming. Technical Report 2007-02, Staatstoezicht op de volksgezondheid. Inspectie voor de Gezondheidszorg, Feb. 2007. http://www.igz.nl/15451/475693/2007-02_Rapport_Preoperatie1.pdf.

Access Control Requirements for Processing Electronic Health Records

Bandar Alhaqbani[1] and Colin Fidge[2]

[1]Information Security Institute,
[2]School of Software Engineering and Data Communications,
Queensland University of Technology, Brisbane, Australia.
b.alhaqbani@isi.qut.edu.au c.fidge@qut.edu.au

Abstract. There is currently a strong focus worldwide on the potential of large-scale Electronic Health Record systems to cut costs and improve patient outcomes through increased efficiency. A number of countries are developing nationwide EHR systems to aggregate services currently provided by isolated Electronic Medical Record databases. However, such aggregation introduces new risks for patient privacy and data security, both by linking previously-separate pieces of information about an individual, and by creating single access points to a wide range of personal data. It is thus essential that new access control policies and mechanisms are devised for federated Electronic Health Record systems, to ensure not only that sensitive patient data is accessible by authorized personnel only, but also that it is available when needed in life-critical situations. Here we review the traditional security models for access control, Discretionary Access Control, Mandatory Access Control and Role-Based Access Control, and use a case study to demonstrate that no single one of them is sufficient in a federated healthcare environment. We then show how the required level of data security can be achieved through a judicious combination of all three mechanisms.

1 Introduction

The healthcare domain—as one of the world's largest hybrid organizations—stands to gain enormously from increased adoption of Information and Communications Technologies. Electronic Health Record (EHR) systems are the latest evolution of healthcare ICT, and countries such as Australia, the United Kingdom and the USA are all working on plans for national EHR systems [9].

An Electronic Health Record is defined by Iakovidis [12] as "digitally stored healthcare information about an individual's lifetime with the purpose of supporting continuity of care, education and research, and ensuring confidentiality at all times". It is a mechanism for integrating healthcare information currently collected in both paper files and Electronic Medical Record (EMR) databases by a variety of separate healthcare providers [16].

Electronic Health Records enable efficient communication of medical information, and thus reduce costs and administrative overheads [5]. Furthermore,

EHRs will help to reduce incidents of medication error—in current healthcare systems, medical data is entered and can be interpreted in inconsistent and possibly ambiguous ways. Moreover, a patient's health records are currently often dispersed over multiple sites with no single healthcare professional having access to all of this data. Nationwide EHR systems aim to solve these problems.

However, to achieve these potential benefits, the healthcare industry must overcome several significant obstacles. Currently, medical information is stored in a variety of proprietary formats using numerous off-the-shelf and custom-built medical information systems. This results in a severe inter-operability problem in the healthcare sector [4].

Also, the security of each patient's medical data is a major issue [15] which, if not addressed in both a technically-sufficient and transparent way, will lose the patient's confidence in and trust of the EHR system. In a worst-case scenario, patients may resorting to falsifying information in an attempt to preserve their privacy, thus affecting the integrity of the stored data and potentially leading to life-threatening situations such as inappropriate medication. Chhanabhai et al. [2] showed in their EHR usability survey that 73.3% of participants were highly concerned about the security and privacy of their health records. Their study indicated that consumers are ready to accept the transition to EHR systems, but only provided they can be assured of the system's security.

Several solutions are available to overcome the security concerns associated with EHR systems. Cryptographic technology, through the use of Public Key Infrastructure [3], allows confidential information to be transmitted safely via an insecure communications medium such as the Internet. On its own, however, cryptography merely handles the security of data transmission and does not address the issue of what kind of data is transmitted, or solve the problem of who has access to the data at the sending and receiving ends.

To do this we need to consider access control mechanisms that limit who can see Electronic Health Records and how they can manipulate them. Access control mechanisms have been through many developments [14] in both academia and industry in order to satisfy the needs of healthcare domains. However, progress to date have not been sufficient to meet the security requirements of a federated healthcare environment [8]. Most of the models developed so far have been designed to satisfy healthcare security requirements in a controlled environment, such as the Electronic Medical Record database maintained within a hospital. By contrast, access control mechanisms for EHRs must be safe for use in open networks, such as the Internet, and with peripheral equipment that was not designed for highly-secure operations, such as a patient's home computer.

Discretionary Access Control (DAC), Mandatory Access Control (MAC) and Role-Based Access Control (RBAC) are well-established access control principles and have been recognized as official standards. Each was designed to overcome limitations found in its predecessor. DAC, the first standard introduced, controls each user's access to information on the basis of the user's identity and authorization [17]. MAC, the second standard introduced, governs access on the basis of the security classification of subjects (users) and objects in the system.

RBAC, the third standard introduced, regulates user access to information on the basis of the activities particular types of users may execute in the system.

In this paper, we demonstrate through case studies that none of these three mechanisms in isolation is sufficient for the privacy and security requirements of Electronic Health Record systems. We then explain how a careful combination of all three access control standards can be used to deliver the essential security requirements of a federated EHR system.

2 Related and previous work

An access control mechanism is intended to limit the actions or operations that a legitimate user of a computer system can perform [17]. This research area has witnessed numerous developments in the last two decades that have resulted in the widespread adoption of three different access control models. In this section we introduce these three models and point out which of their known limitations would apply in the healthcare domain.

2.1 Discretionary access control

Discretionary Access Control is a means of restricting access to objects based on the identity of subjects and/or groups to which they belong [7]. The controls are discretionary in the sense that a user or subject given discretionary access to a resource is capable of passing that capability along to another subject. The identity of the users and objects is the key to discretionary access control. DAC policies tend to be very flexible and are widely used. However, DAC policies are known to be inherently weak for two reasons: granting read access is transitive and DAC policies are vulnerable to "Trojan horse" attacks [10, 7].

DAC policies are commonly implemented through Access Control Lists (ACLs) and 'owner/other' access control mechanisms, but these mechanisms are difficult to manage because addition and deletion of users or data objects requires discovery and treatment of all dependent entries in the DAC matrix.

In an Electronic Health Record system the access control requirements are more complex than allowed for by Discretionary Access Control because the data in an EHR is nominally 'owned' by the patient [9], but is also updated by healthcare professionals, and is stored on infrastructure belonging to healthcare providers and regulators. Indeed, a DAC model could create new security problems due to the patient's mismanagement of their own records [7].

2.2 Mandatory access control

A Mandatory Access Control policy, which is known to prevent the "Trojan horse" problem [7, 17], means that access control decisions are made by a central authority, not by the individual owner of an object, and the owner cannot change access rights. The need for a MAC mechanism arises when the security policy of

a system dictates that protection decisions must not be decided by the object's owner, and the system must enforce data protection decisions [7].

Mandatory Access Control typically occurs in military-style security. Usually a security labeling mechanism and a set of interfaces are used to determine access based on the MAC policy. For example, a user who is running a process at the *Secret* classification level should not be allowed to read a file with a label of *Top Secret*. This is known as the "simple security rule", or "no read up". By contrast, a user who is running a process with a label of *Secret* should not be allowed to write to a document with a label of *Confidential*. This rule is called the "$\star$-property" or "no write down". Multilevel security models such as the Bell-La Padula Confidentiality [6] and Biba Integrity [1] models are used to formally specify this kind of MAC policy. Nevertheless, unintended information transfer can occur in systems using MAC through covert channels, whereby information of a higher security class is deduced indirectly by intelligently combining information visible to a lower security class [10].

Applying Mandatory Access Control mechanisms in an EHR environment is likely to be very difficult due to the huge number of users who participate in those systems, the wide range of data types, and the desire to give patients ownership and (partial) control over their own medical records. Nevertheless, implementing some form of MAC policy is inevitable in an EHR system, since medical authorities must be ultimately responsible for assigning access rights [5].

2.3 Role-based access control

Role-Based Access Control decisions are based on the roles that individual users have as part of an organization. Users take on assigned roles (e.g. doctor, nurse or receptionist in our case). Access rights (or permissions) are then grouped by role name, and the use of resources is restricted to authorized individuals [10]. Under RBAC, users are granted membership into roles based on their competencies, credentials and responsibilities in the organization. User membership in roles can be revoked easily and new operations established as job assignments dictate. This simplifies the administration and management of permissions since roles can be updated without updating the permissions for every individual user. Moreover, use of role hierarchies provides additional advantages since one role may implicitly include the operations associated with another role. Also, RBAC can satisfy the "least privilege access" requirement [17], which involves granting the minimum set of privileges required for individuals to perform their job functions. Separation of Duty (SoD) can be incorporated into the RBAC model [18] to ensure that a user is not allowed to execute two roles simultaneously, as per the organization's policy.

Role-Based Access Control has gained a lot of attention in healthcare security research thanks to: its ability to provide practical fine-grained access policy administration for a large number of users and resources; it being a neutral policy; and its support for the 'need-to-know' security principle.

However, some access request evaluations are complex, due to the need to consider other contextual parameters in the evaluation phase. To overcome this

problem, the Contextual RBAC model adds contextual parameters (e.g. time and location) to the RBAC model [19]. Nevertheless, even Contextual RBAC is insufficient to support the dynamic permission assignments that are needed in the healthcare domain, so Motta et al. [13] extended the model further so that permission assignment is based on specific evaluation mechanisms using contextual attributes that are available at access time, and Wilikens et al. [19] used a trust level as a measurement to assign permissions.

Unfortunately, this extended process would add yet more complexity to an EHR system — which requires establishing a connection between the EHR system and the Hospital Information Systems (HISs) that are responsible for handling administrative work within a hospital — in order to collect those contextual attributes which are not immediately visible, for example, the requestor's current medical role (e.g. as a doctor in an emergency department).

3 Healthcare access control requirements

In the previous section we reviewed the capabilities of Discretionary Access Control, Mandatory Access Control and Role-Based Access Control. In order to better understand what kind of access control solution is needed for an Electronic Health Record system, we summarize in this section the specific access control requirements peculiar to EHR systems, illustrated by a small case study, and review the weaknesses of the existing mechanisms in this situation.

A control mechanism for Electronic Health Record access must satisfy all EHR participants' needs, i.e. patients, medical practitioners and medical authorities. Each participant needs to access certain fields of the health record in order to carry out his job. Also, the various participants need the ability to set specific access controls over the record. The following privacy and security requirements have been identified as crucial to healthcare environments:

1. Each healthcare unit should have the freedom to design its own security policy and to enforce it within its domain [15].
2. Healthcare providers (e.g. General Practitioners) should have the flexibility to arbitrarily define the security of a particular document if so required.
3. Patients should have the right to have control over their own health records, including whether or not to grant access to certain medical practitioners [15].
4. Patients should be able to hide specific items of information contained in their health records from selected medical practitioners.
5. Patients should have the ability to delegate control over their health records to someone else under certain conditions (e.g. mental illness).
6. Managing access control policies should be an easy task, in order to ensure that the system is used and to preserve trust in the system.
7. It is important that legitimate uses of health records are not hindered, e.g. overall system availability service levels, and overriding 'need-to-know' data access requirements in emergencies.

Ensuring each patient's privacy and data security is vital for an Electronic Health Record system. Unlike paper-based models, where an exposure or intrusion is confined to a single document or file, a federated EHR system creates the possibility of a patient's entire medical history being compromised by a single action. However, each of the traditional access control models, reviewed in Section 2, can satisfy only some of the above-listed requirements.

To see the access control weaknesses inherent in these previous models, consider the following scenario:

> Frank prefers to go to two General Practitioners, Tony and Karen. Frank has two sensitive fields in his Electronic Health Record, describing mental illness and sexual issues, respectively. Frank is happy to let Tony have access to his EHR, including the relevant data field within his sexual record, but he wants to hide a data field within his mental illness record from Tony. On the other hand, Frank will allow Karen to access his EHR, including his entire mental illness record, but not the sensitive data field within his sexual record. Apart from these two GPs, Frank won't allow anyone to access the sensitive data fields in his mental or sexual health records. In addition, Frank's father John suffers from Alzheimer's disease, so Frank must manage the access control rights to his father's EHR.

Even this simple and unremarkable scenario creates problems for each of the traditional access control policies, as explained below.

Discretionary Access Control: To use a DAC model we first need to know who owns the Electronic Health Record because DAC assumes that the owner of the data is the one who controls access to it. However, in healthcare an EHR is partially owned by each of the patient, medical practitioner and medical authority [11], immediately creating an issue with respect to ownership. Furthermore, assuming that Frank has ownership of his EHR, he could nominate and grant access to his trusted/preferred medical practitioners (Tony and Karen), but it would be a difficult task for Frank as a non-expert to identify the specific medical data that is needed by each GP to do their job. The 'need-to-know' principle is required here, and in order to have it Frank is required to know the information that is needed for each medical practitioner and then set the access controls accordingly. By granting patients such control over their records, we may hinder the legitimate use of the EHR and, most likely, create another security problem due to the patient's mismanagement of their records. However, delegation of access control can be implemented easily in a DAC model. Since Frank's father owns his EHR, he can delegate access control to his son.

Mandatory Access Control: In a MAC model, Frank won't have any sort of control, because the Electronic Health Record system will be responsible for setting the security labels for users and EHR data objects. Therefore, Frank can't express his access control wishes over his EHR. Also, the 'need-to-know' principle can't be fully achieved here either, even if we apply a security level hierarchy. It's possible that two users might have the same security clearance (e.g. Tony and Karen), but should have different access permissions over a certain data object (e.g. mental health data). In the MAC case, we can't assign more than

one security label for the same data object, so providing selective access to data objects is difficult. Moreover, there is no ability to delegate access control because patients have no control over their EHRs.

Role-Based Access Control: In an RBAC model, the 'need-to-know' principle can be satisfied by defining the permissions/operations that are required by a specific medical role, and this process could be done by an appropriate medical expert. However, in this situation Frank won't be able to hide his sensitive medical fields as he won't have any control over the permission assignments. In order to allow Frank to express his wishes, the information security officer must allow Frank to modify the permissions, roles, user-role and role-permission assignments. Frank would need to create three roles in order to satisfy his needs, which would become an unacceptably time-consuming and complicated task for most patients and is likely to lead to a conflict of access control settings. Delegation of roles in RBAC is permitted if the security officer would allow Frank's father to delegate his roles to his son. Generally, RBAC seems a better choice than DAC and MAC, though it is still not an adequate solution for EHR system security.

In summary, it is clear than none of the existing models is adequate on its own, but that each of them has some feature which is essential to an EHR security model. DAC allows patients to control which data can be seen by particular medical practitioners, MAC allows the medical authority to control access to specific kinds of data, and RBAC allows access rights to be associated with certain medical roles.

4 A combined access control protocol

Although the access control requirements for Electronic Health Records cannot be satisfied by any one access control model alone, we contend that a careful integration of all three existing models is sufficient. Combining existing models, rather then developing an entirely new one for healthcare, allows us to take advantage of the well-understood properties and established implementations for these models.

4.1 Overview of the combined protocol

In the combined model access to a particular Electronic Health Record data item is granted only if it satisfies all three policies. The challenge is to determine where and how each of the access control constraints is introduced.

The basis for our combined protocol is shown in Fig. 1. An Electronic Health Record schema is shown where each EHR field has two MAC-based security labels: one is assigned by the patient and the second is assigned by the medical practitioner. These labels are used to express the sensitivity class of the data field. Also a DAC-style Access Control List (ACL) is maintained by the patient, whereby the patient nominates his/her preferred/trusted medical practitioners and sets the security clearance for each of them. This security clearance allows

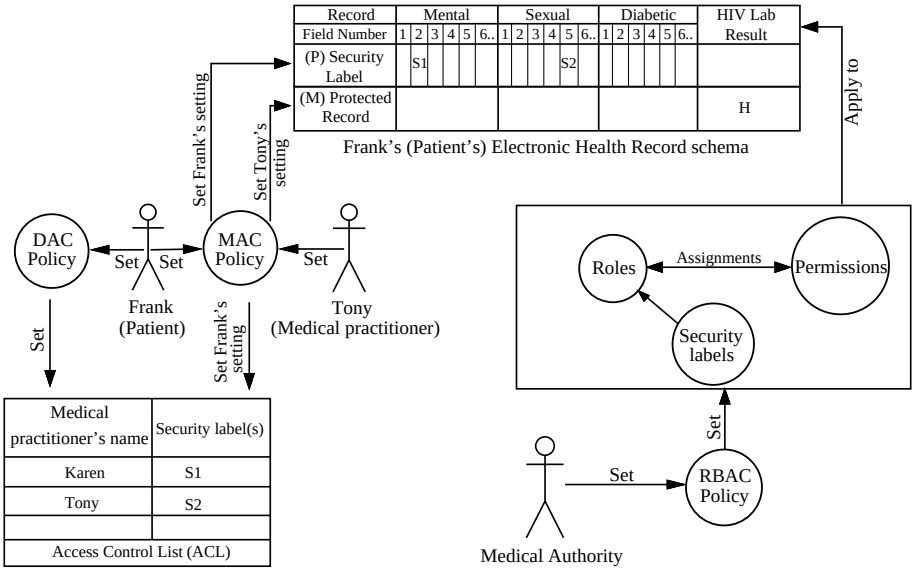

Fig. 1. The logical structure of the combined access control protocol

the medical practitioner to access sensitive data that may not be allowed for other medical practitioners. Access to EHR fields is further restricted by overall RBAC-based access control managed by the medical authority.

4.2 Maintenance and enforcement of access control constraints

Each of the participants in the EHR system (patients, medical practitioners and medical authorities) needs to maintain some aspect of the combined access control policy, and is constrained in what information they can view as a result. In this section we describe the sequence of events needed to do this.

We start with the patients' access control requirements, where the patients want to decide who is authorized to access their Electronic Health Records, to determine what is the sensitive information in their EHRs, and who is authorized to access it. These requirements are satisfied by executing the following steps using the DAC and MAC interfaces in our combined access control policy:

1. The patients nominate the names of specific practitioners who they trust, and this is done through the DAC interface in Fig. 1 to construct the Access Control List (ACL).
2. To categorize data fields as sensitive/protected information, the patient needs to assign security labels to these data fields by using the MAC interface to update the patient's Electronic Health Record schema.
3. To allow specific medical practitioners to gain access to security-classified data fields in the patient's EHR, the patient, via the MAC interface, assigns

the same security label of the sensitive data field to the authorized medical practitioners' ACL.

In practice, however, we do not suggest using the "no read up" and "no write down" rules that are introduced in MAC because it would be too complex a task for most patients to keep track of the transitive relationships introduced by a full hierarchy of security levels. Instead patients should just be presented with simple access/no-access settings.

Medical practitioners, as EHR consumers, have certain access control requirements that are important. Medical practitioners need to:

- access all the information that is required to fulfill their medical role in normal scenarios (e.g. a standard consultation with a GP), unless the patient has excluded that practitioner from accessing the particular data field;
- access all the information that is required in emergency cases regardless of the patient's access control settings; and
- hide some medical information from the patient.

Medical practitioners' access control requirements are also satisfied here. The following steps show how these requirements are met:

1. The Medical authority defines roles, permissions and role-permission assignments via the RBAC interface. This process is done by domain experts who know the access requirements for each medical role. Therefore, the 'need-to-know' principle is achieved and medical practitioners' access needs will not be limited unless the patient has set some access control restriction through either the DAC or MAC interface.
2. Since RBAC can incorporate contextual attributes into role assignments, it would be possible for a medical practitioner to have an access role as a GP in a day clinic or as a GP in an emergency department. To allow the GP in an emergency department to access security-classified data records, the RBAC policy would assign a security label to these critical roles to allow them access to secure data. In an emergency case, the DAC constraints are not evaluated, due to the fact that the patient won't be able to know who the attending medical practitioners will be in an emergency.
3. To hide some medical information from the patient, the medical practitioner, through the use of the MAC interface, sets protection labels for these fields which hide the existence of such data in the patient's EHR.

Finally, the medical authority in charge of managing the Electronic Health Record system acts as the 'security officer' in the RBAC interface. It defines roles and permissions, and controls the assignment of permissions to roles in order to associate specific medical roles with the information needed to fulfill them.

4.3 Motivational example revisited

In this section we revisit the motivational scenario from Section 3 to see how our access control protocol would satisfy Frank's wishes.

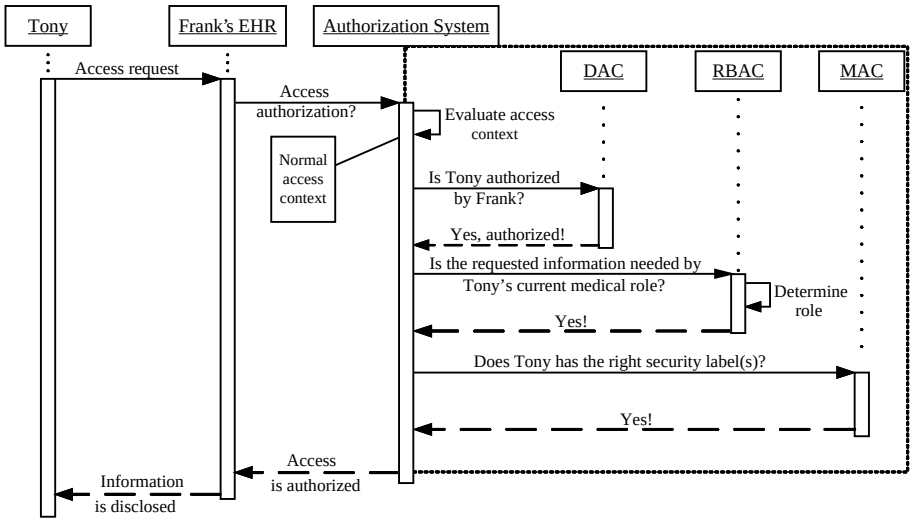

Fig. 2. The authorization evaluation process in the motivational example

1. Frank will classify the mental and sexual data fields of concern as 'sensitive' information by setting security labels S1 and S2 for each record, respectively.
2. He will nominate his preferred GPs Karen and Tony to access his EHR. As Frank is happy to allow Karen to access his sensitive mental data field, he will assign the S1 security label to her, which means that she is authorized to access any sensitive information that has an S1 label, in addition to her authorized access as per her medical role. For the same reason, Frank will assign the S2 security label to Tony which will allow him to access the sensitive sexual data field.
3. When Tony requests access to Frank's Electronic Health Record, to see his sexual history, the following access evaluation occurs (Fig. 2):
   (a) Evaluate access context, 'normal' or 'emergency'. If it's an emergency go to step 3c, otherwise continue.
   (b) DAC policy: Does Frank authorize Tony to access his EHR?
   (c) RBAC policy: Determine Tony's current medical role (e.g. day clinic GP, Emergency doctor) based on current contextual conditions.
   (d) RBAC policy: Is the requested information needed by Tony's current medical role?
   (e) MAC policy: Is Tony cleared to access this sensitive record?
   (f) Tony is granted access to Frank's sexual data only if his access request passes all the steps above.

Also, as Frank needs to take responsibility for his father's Electronic Health Record, the following actions can be performed.

1. Frank's father John needs to delegate control over his EHR to Frank through the use of the DAC interface.

2. Frank can now set the access rights to his father's EHR.

As well as these static assignments, we also need to consider temporary changes to access requirements. For instance, assume that Tony asks to see Frank's mental health record because he thinks that Frank's sexual issue is affected by some mental illness. This means that Frank must give Tony temporary access to his sensitive mental data field.

1. Frank will grant Tony another security label, S1.
2. Tony now has two security labels S1 and S2 from Frank, which means that he is authorized to access both of Frank's sensitive data fields contained in his sexual and mental health records.
3. After the consultation, Frank can revoke this permission by deleting label S1 from Tony's profile.

On the other hand, a medical practitioner may need to change the status of certain fields without involving the patient. For instance, assume that Tony asks Frank to take a blood test which turns out to be positive for HIV. Given Frank's mental state, Tony would prefer to hide the pathology results until Frank's next in-house consultation.

1. Tony assigns a 'hide' flag to the HIV lab result field in Frank's EHR, so that Frank can't see any information contained in that specific field.
2. However, this information can be seen by Frank's authorized medical practitioners, such as the blood bank to which Frank regularly donates.

5 Conclusion and future work

Emerging plans for national Electronic Health Record systems raise new concerns about patient privacy and data security, by merging medical records that were previously kept separate and by making them accessible through single access points. None of the three standard access control models, Discretionary Access Control, Mandatory Access Control and Role-Based Access Control, are adequate for an EHR system in isolation. Nevertheless, we have explained how a careful combination of all three access control models can provide the security functionality needed for an EHR system.

At the time of writing we are assessing the security issues associated with a prototype Service Oriented Architecture for healthcare data. Our goal is to determine whether such an 'application-oriented' networking environment can be used to implement the combined access control protocol described above.

Acknowledgements We wish to thank the anonymous reviewers for their many helpful suggestions. This research was supported in part by Australian Research Council Linkage-Projects grant LP0776344.

References

1. K. J. Biba. Integrity Considerations for Secure Computer System. Technical report, Mitre Corporation, 1977.
2. P. Chhanabhai and A. Holt. Consumers are Ready to Accept the Transition to Online and Electronic Records if They can be Assured of the Security Measures. *Medscape General Medicine*, 9(1), 2007.
3. L. Demuynck and B. De Decker. Privacy-Preserving Electronic Health Records. In *Communications and Multimedia Security*, volume 3677 of *Lecture Notes in Computer Science*, pages 150–159. 2005.
4. M. Eichelberg, T. Aden, Riesmeier J., A. Dogac, and G. Laleci. A Survey and Analysis of Electronic Healthcare Record Standards. *ACM Computing Surveys*, 37(4):277–315, 2005.
5. HealthConnect Business Architecture, version 1.0., 2003.
6. D. E. Bell and L. J. LaPadula. Secure Computer Systems: Unified Exposition and Multics Interpretation. Technical report, Mitre Corporation, 1976.
7. D. Ferraiolo, D. Kuhn, and R. Chandramouli. *Role-Based Access Control*. Artech House, 2003.
8. B. Finance, S. Medjdoub, and P. Pucheral. Privacy of Medical Records: From Law Principles to Practice. In *Computer-Based Medical Systems, 2005. Proceedings. 18th IEEE Symposium on*, pages 220–225, 2005.
9. D. Tracy Gunter and P. Nicolas Terry. The Emergence of National Electronic Health Record Architectures in the United States and Australia: Models, Costs, and Questions. *Journal of Medical Internet Research*, 7(1):e3, 2005.
10. V. Hu, D. Ferraiolo, and D. Kuhn. Assessment of Access Control Systems. Technical report, National Institute of Standards and Technology, September 2006.
11. L. Iacovino. Trustworthy Shared Electronic Health Records: Recordkeeping Requirements and HealthConnect. *Journal of Law and Medicine*, 12:40–60, 2004.
12. I. Iakovidis. Towards Personal Health Record: Current Situation, Obstacles and Trends in Implementation of Electronic Healthcare Record in Europe. *International Journal of Medical Informatics*, 52(1–3):105–115, 1998.
13. G. Motta and Sérgio Shiguemi Furuie. A Contextual Role-Based Access Control Authorization Model for Electronic Patient Record. *IEEE Transactions on Information Technology in Biomedicine*, 7(3):202–207, 2003.
14. J. Park and R. Sandhu. Towards Usage Control Models: Beyond Traditional Access Control. In *SACMAT '02: Proceedings of the 7th ACM symposium on Access Control Models and Technologies*, pages 57–64, New York, USA, 2002. ACM Press.
15. P. Ray and J. Wimalasiri. The Need for Technical Solutions for Maintaining the Privacy of EHR. In *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE*, pages 4686–4689, 2006.
16. W. Rishel, T. Handler, and J. Edwards. A Clear Definition of the Electronic Health Record. Technical report, Gartner, 2005.
17. R. Sandhu and P. Samarati. Access Control: Principles and Practice. *IEEE Communications Magazine*, 32(9):40–48, 1994.
18. R. Simon and M. Zurko. Separation of Duty in Role-Based Environments. In *IEEE Computer Security Foundations Workshop*, pages 183–194, 1997.
19. M. Wilikens, S. Feriti, A. Sanna, and M. Masera. A Context-Related Authorization and Access Control Method Based on RBAC: A Case Study from the Health Care Domain. In *Proceedings of the 7th ACM symposium on Access control models and technologies*, pages 117–124. ACM Press, 2002.

Learning Business Process Models: A case study

Johny Ghattas, Pnina Soffer, Mor Peleg
Management Information Systems
University of Haifa, Carmel Mountain 31905, Haifa, Israel
Email: GhattasJohny@gmail.com, SPnina@is.haifa.ac.il, Peleg.Mor@gmail.com

Abstract. Learning how to improve business processes is an evolutionary process that must be managed as other business processes (BPs) are managed in modern organizations. The proposed model – the learning process model (LPM) – suggests a closed-loop-model approach applied to a generic process model (GPM), which is a formal state-based and goal-based approach to process modeling. LPM strives to establish a learning process by (1) identifying goal and soft-goal states of the initial process model, (2) identifying exceptional states and incomplete state definitions at runtime, and (3) adapting automatically the process model according to the discovered states. Modifications provided by the learning process may be sufficient or may need to be complemented by non-automatic changes, when unacceptable business situations arise. The learning process also aims to adapt the current process model to possible technology, specific domain (e.g., clinical procedures at specific institutions), environmental requirements (e.g., regulations and policies), and process innovations. We demonstrate the application of LPM to a vaccination process.

Keywords: Learning, business process model, generic process model, clinical guidelines, exceptions, process flexibility, process adaptation, goals, soft-goals.

1. Introduction

In a dynamic business environment, business processes (BPs) need to be changed continuously [1-4] without affecting the production of the expected business values. The continuous business environment change, the shortening of required service time to market, the increasing number of inter and intra-organization integrations and the early adoption of new business technologies makes it impossible to fully-represent business processes during their conception time.

In this research, we postulate that it is possible to substitute the practice of fully representing business processes by a flexible adaptive form of designing, implementing, and managing business processes, through a business process learning approach. This approach would allow making a partial (minimal) definition of the business processes that would enable the organization to launch the required services with minimal time to market and seize business opportunities. Once the required

business initiative is launched and operating, data regarding the process execution and the extent to which goals and soft goals are attained are collected and analyzed, and deviations between the currently defined process model and the actual business process are detected. This forms the basis for learning and adapting the process based on the day by day process enactment experience.

Such an approach is of particular importance in health-care processes, which may change on the fly in adaptation to specific patients, or change over time as a result of the availability of new knowledge. The clinical guidelines modeling research field provides several approaches for medical guidelines automation, discovery and even adaptation [5-8].

This paper envisions learning in business processes, and constitutes a first step towards the design of automated processes that (1) define and track suitable parameters about the process execution and (2) learn how these processes should improve and update the process model accordingly. We base our approach on the Generic Process Model (GPM) [9, 10] which is a formal process specification, suitable for our purposes due to its explicit representation of goals. The paper illustrates a potential model for the business process learning through a medical immunization case study.

2. Business Process learning model – a proposal

We postulate that a process model needs to relate the process goals to the workflow required to accomplish them. Such a relation is necessary in order to evaluate whether the process execution is attaining its desired outcome (i.e., goals) and the performance of the process execution (i.e., soft-goals). In order for our model to be as formal as possible, we base it on the Generic process Model (GPM).

2.1 Using a Formal BPM – the Generic process model (GPM)

The focus of analysis in GPM is a domain, which is a part of the world. We describe the behavior of the domain using concepts from Bunge's ontology [11, 12] and its adaptation to information systems [13, 14]. A domain is represented by a set of state variables, each depicting a relevant property of the domain and its value at a given time. We view a successful process as a sequence of unstable states of the domain, leading to a stable state, which belongs to a set of states that reflect the process goal. An unstable state is a state that must change due to actions within the domain (an internal event) while a stable state is a state that does not change unless forced to by action of the environment (an external event). Internal events are governed by transition laws that define the allowed (or necessary) state transitions (events).

It is possible to define the projection of a process over a sub-domain, where the set of state variables addressed by the law is a subset of domain state-variables. Then, all transitions outside the sub-domain are considered external events, and the sub-domain may be in a stable state while the process is active in other parts of the domain.

The process goal, as addressed by GPM, is the state achieved by the process. However, the goal concept is sometimes used also to describe business objectives.

GPM distinguishes process goals from *soft-goals*, which are defined as an order relation on goal states [10]. In other words, soft-goals relate to the desirability of possible states in the goal set (all meeting the condition that terminates the process) according to defined business objectives. These establish a ranking (order) among the goal states. For example, assume the goal of a process is a set of states where some medical treatment has been given to a patient, but a lower level of the patient's blood pressure following this treatment is considered better than a higher one.

Finally, GPM entails criteria for assessing the *validity* of a process, namely, its ability to achieve its goal [10]. It enables the analysis of a process to identify causes for invalidity and suggests appropriate redesign actions to eliminate these causes.

2.2 Business process learning - definition

While a process can reach its goal states through different paths, these paths may attain different soft-goal levels. In addition, while a particular path may improve a specific soft-goal it might simultaneously worsen another. Based upon the GPM process definition, the result of process instance executions may be categorized to two main categories:

(1) Valid process instances which attain some process goal state.
(2) Invalid process instances which do not reach any process goal state and terminate into exceptions (exceptions occur when a process attains an unlawful state during its execution and cannot reach its goal).

The category of valid process instances may be further divided into an undefined number of sub-categories depending on attained levels of soft-goals. It is needless to say that a process path that leads to better soft-goal levels is preferred to others that lead to lower levels of soft-goals. Hence, it is clear that an organization must strive to improve continuously its capabilities to select better process paths in order to attain better soft-goal levels as well as fewer exception occurrences in runtime, namely, fewer instances where the process fails to achieve its goal. In parallel, the organization must strive to adapt (modify) its BP models because of new knowledge generated in the environment. *Business process learning* is the organization's capability to improve path selection through experience acquired from executing the business process, measuring the attained levels of soft goals and the exceptions rate.

2.3 Business process learning- a process-based approach

We postulate that a business process learning model can be established analogously to control systems theory, which bases the continuous improvement of the behavior of a controlled system through a closed-loop system-control model [15]. The closed-loop model is based upon a real-time feedback-loop that continuously modifies the system behavior in order to minimize the output error attained during system runtime. A thorough discussion of control systems can be found in [15].

Adapting the model to BPM, at each process enactment, implies that:

(1) Soft-goals are measured and following their scores, the process model is evaluated to identify what specific segments of the state flow caused the improvement or worsening of each soft-goal.
(2) Whenever exceptions occur they are analyzed and learned lessons are used to recommend process model changes
(3) Once soft goal scores and exceptions are identified, the process law may be changed accordingly, so similar state flow will be repeated or avoided in future instances of the process. The updates may require changes in state definitions (i.e., modification of the set of relevant state variables, addition/modification of predicates defining sets of states, and/or transformation definitions).

A closed-loop BP learning approach leads to the definition of the following learning process:

(1) During process instance execution, a set of state variables that would enable the evaluation of the attained level of soft-goal and the occurrences of exceptions are collected.
(2) Process instance soft-goal levels are evaluated.
(3) Collected process datum of the specific process instance, together with soft-goal levels and exception occurrence indicators are stored.
(4) The current process instance is compared to past experiences and assigned a relative score.
(5) Future executions would use the collected information and score to select in runtime the best known path.

2.4 Learning Process model (LPM) Assumptions

For any formal conceptual model to be complete and valid, the ontological assumptions of the model need to be made explicit.

The proposed LPM relies on the following assumptions:

(1) The initial process model is valid, as far as we know (i.e., it was set in a way that is meant to attain its goals considering an expected set of possible external events independently of the process learning capability).
(2) Process mining capabilities do not affect process run time nor process performance.
(3) Process soft goals are known a-priori and are measurable through the process mining capabilities.
(4) Learning is based on the gaps between desired goals and actual execution of process instances. The process model is modified by: (a) comparing actual soft goals measurement to historical values of the process soft goals; (b) analyzing the current state flow as compared to past occurrences in order to explain the accomplished levels of soft-goals; (c) drawing required process model changes from exceptional process instances.

2.5 LPM components

We establish the following postulates:

Postulate 1: In order for the overall business process to be a learning process, it should include three main sub-processes: Acting (A) process, Documenting (D) process, and Learning (L) process, as described below.
Postulate 2: These three processes are (in terms of the GPM) projections of the business process executed by the organization upon the respective sub-process domains (Acting, Documenting and Learning process domains).
Postulate 3: The A, D, and L processes interact through a set of well-defined *commitments*, as explained below. In addition, the external environment can affect these processes.
Postulate 4: The overall BP needs to adapt according to environment inputs.

The respective five model components are hereby described:
Component 1: An acting sub-process – the process that acts in order to accomplish the goals and soft goals of the process.
Component 2: A documentation sub-process – the process that collects the necessary data from the acting process for three main purposes: (1) conditioning next actions in the current acting process instance depending on data collected in previous process instance steps and data collected from past process instances; data may also be collected from external business processes (the fifth component of the model- see below); (2) affecting actions in other process instances (current and future ones); (3) providing learning processes with data collected and processed during the process enactments (i.e., soft-goal measures and exception details).
Component 3: A learning sub-process – the process in charge of adapting the business process model. It analyzes the collected data, produces required changes to support incurred exceptions, models needed changes, and introduces changes to the BPM. Note that each one of these sub-processes has its own goals and soft-goals.
Component 4: Inter-process dependencies: The commitments between the internal business sub-processes (learning, acting, documenting) is the basis for defining a valid overall business process. The acting process commitments are: (1) to provide necessary data for documentation process; (2) to execute according to the business process model that may be changed by the Learning sub-process.
The documenting process commitments are: (1) to collect necessary data for the acting process control/decision points; (2) to provide the acting process with data collected from previous process steps, past process instances, and data collected from external business processes; (3) to collect data for the learning sub-processes.
The learning process commitments are: (1) to provide/execute necessary changes to the business process model. Note that changes (both to D and A processes) have two major sources: needed adaptations following soft-goal assessments and exceptions detected; (2) to provide visibility/traceability of changes, structured process history, reports needed for human intervention.
Component 5: External (Human) processes & commitments: These are processes that affect the L, A, and D processes through inputs that are generated by the external environment**.** External processes are of two kinds: (1) modeling new cases, processes, and introducing innovation, when new medical knowledge is available (e.g., new drug, new available immunization); (2) manual management for handling exceptions. These are scenarios where the acting, learning, or documenting processes fail to continue their executing due to anomalies or unexpected situations. In such cases,

automatic learning is not always feasible, and an external entity (e.g., a human) may intervene to correct the situation.

3. LPM Illustration through a case STUDY

We use a clinical process based upon the guidelines for immunizations provided by the Institute for Clinical systems improvements (ICSI, [16]) as a case study. We start by presenting a hypothetical local version of the generic algorithm that addresses flu vaccination and is adapted to the workflow and regulation of a particular implementing healthcare institution. Next, we map the local process flow to our learning model and demonstrate how monitoring electronic medical record (EMR) data can be used to follow whether the executed process attains its goals or reaches undesired states, and how we can learn, that is, modify the process model in order to improve the extent to which it attains its goals or avoids undesired states.

3.1 Defining a local version of the immunization process

Fig. 1 shows a hypothetical local flu immunization process that follows the ICSI clinical algorithm [16]. When the patient is vaccinated for the first time in her life, the vaccine is provided in two portions, which are to be administered in two separate visits that are spaced one month apart.

3.2 Identification of the Flu Vaccine Business Process

We map the algorithm represented above into the components of the GPM process model – goals, soft-goals, states, intermediary states, and laws.

Identification of process goals and sub-goals:

Process goals: **(G1)** "Identify an eligible child"
(G2) "Provide vaccinations to an eligible child"

Soft-goals: **(SG1)** "Do our best to make the parent accept vaccinating his child. Parent Refusal to vaccinate is an undesirable outcome". The score of this soft-goal is evaluated at the reached goal state; we consider it having a binary value: {desirable, undesirable}.

Goals and soft-goals are represented by process states in GPM. Although the process flow of Fig. 1 does not have states, only activities, we use the convention that when a process has completed an activity it is in a state named by the activity, and this state remains until execution of a new activity begins. In this way, we associate goal and soft-goal states with outputs of different steps in the algorithm, as summarized in Table 1 and as marked in Fig. 1. Note that since goal states are stable states, the goal state corresponding to G1 is step 16 in the process model and not step 3. Note also

that the soft-goal in this case is of a discrete nature (i.e., goal state desirable or undesirable), whereas in other cases it may relate to continuous values.

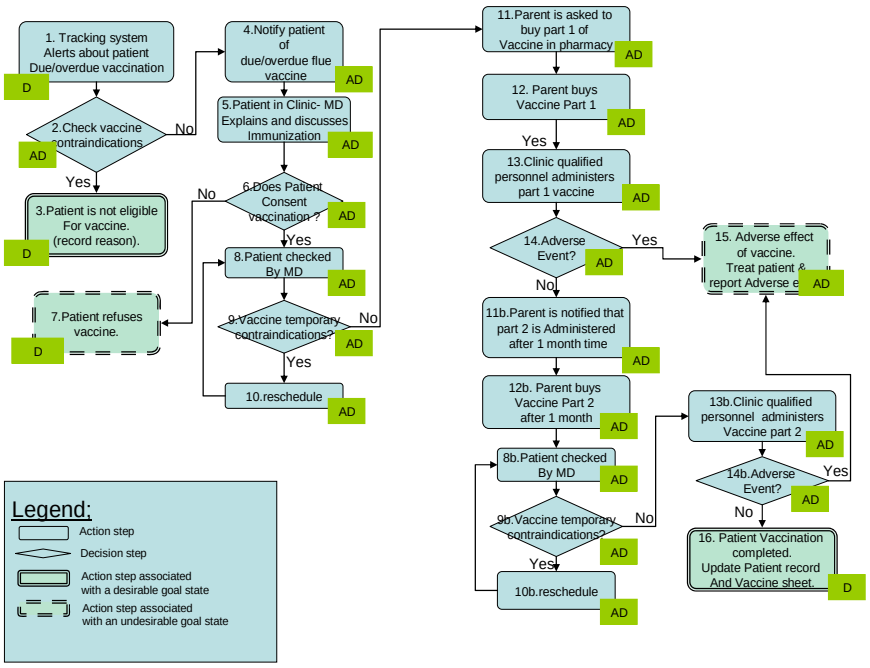

Fig. 1. Local version of the flu vaccination algorithm (first time vaccination). Each step has been identified with the sub-processes: A-Acting, D-Documenting, L-Learning. We identified soft goals: Undesirable Goal states (steps 7, 15) and Desirable Goal states (steps 3, 16)

Table 1. Goal mapping to Flu vaccination algorithm steps (algorithm represented in **Fig. 1**.

Goal states	Goal state definition	Desirable/Undesirable (soft-goal accomplishment level)	Step Outputs
G1	Identify an eligible child	Desirable	16
		Desirable (or normally expected)	3
G2	"Provide flu vaccination to an eligible child".	Desirable	16
		Undesirable	7
		Undesirable	15

Process states identification

Mapping intermediary process states is done in the same way as process goal states. Each state is identified by a set of variables, as we demonstrate in Table 2 (we present here only state variables that are changed within each state, not the whole set of variables associated with all process states). Note that the process has several states

that represent goal states (S2, S6, S11 and S14); process soft-goals are evaluated through the desirability of these goal states (see Table 1).

Table 2. Business Process states mapping for the vaccination algorithm. State types may be S-stable, U-Unstable or G-Goal. The state description indicates (in parentheses) the steps whose outcome corresponds to these states. Note that the outcome of several steps correspond to the same state. In addition, not all steps necessarily cause a state transition.

ID	**State name (corresponding step outcome)**	**Type**	**State variables update**	**Next state**
S0	Initial state (1).	S	--	S1, S2
S1	Checking Vaccine contraindications (2).	U	Eligible flu V. = Yes.	S3
S2	Contraindications present (3).	G	Eligible flu V. = No.	--
S3	Patient notified-overdue vaccine (4)	S	Flu V. Status= eligible; Notified=Yes.	S3, S4
S4	MD discusses Vaccine with Patient (5)	U	Flue V. MD Check=done.	S5, S6
S5	MD checks Patient (6,8,8b)	U	Flue V. MD checking = Yes.	S7, S8,S13
S6	Patient refuses vaccine (7).	G	Flu V. Status = Refused.	--
S7	Parent asked to buy part 1 of Vaccine (11,12).	S	Flu V. Status= "waiting - part 1"	S7, S9
S8	Temporal contraindications present (9,9b,10,10b)	U	Flu V. Status= "rescheduled-contraindications".	S7, S8, S14
S9	First part vaccine administered (13)	U	Flu V. Status= "1st part administered".	S10, S11
S10	Patient Notified- part 2 within 1 month (11b).	S	Flu V. Status= "waiting for part 2".	S10, S12
S11	Adverse event present reported (14,14b,15)	G	Adverse event = Yes; Adverse event report=<…>.	--
S12	Parent buys part 2 Vaccine (12b)	U	Flu V. Status= "Prepared for part 2".	S5
S13	Part 2 vaccine administered (13b)	U	Flue V.Status= "Part 2-completed".	S11, S14
S14	Flue vaccination completed (16)	G	Flu V. Status= "completed".	--

Our Learning approach is based on collecting data from process instances and evaluating the process execution based on them. In clinical applications, the electronic medical record (EMR) may be used for documenting patient-related process-flows within the clinical system, providing a full tracking of the process state. Obviously, the EMR should include all relevant state variable data of the current and past process states. This has two major outcomes:

(1) The EMR becomes our major data base not only for patient data, but also for process flow information.
(2) We establish important foundations for a formal definition of the data required for process state tracking.

Note that data to be included in the EMR need to reflect not only the current state of the process but also enable the user to have a clear view of the process history. This is a challenge that our approach is capable of solving through the described state mapping- the inherent state variable based state definition. The described example shows how we can assure process tracking by mining the specified state variable data.

Process Law specification:

The process law specifies the possible transitions within the process flow. We mapped the possible transitions for each state in Table 2-"Next state" column. Transitions may be triggered by internal events (i.e., part of the process model) or external events (i.e., events that are generated from outside the process domain). According to GPM, internal events are sufficient for triggering a state transition from an unstable state, whereas a transition from a stable state requires an external event. In Table 2, we establish the state type for each state ("Type" column).

3.3 LPM in the Case Study

We can now demonstrate how learning can be performed.

Outcome data assessment: After the first year of executing the local process, the following conclusions were drawn at the implementing institution:

(1) The number of parents that were willing to vaccinate their children is high (state S5 reached).
(2) The number of patients that completed their vaccination was smaller than those who were willing to vaccinate. This was judged by an exception that occurred along different instances: the process did not reach a goal state, but remained stuck in step 12b (state S12).

Identifying possible causes through execution and external data analysis: Further investigation showed that some of the parents who consented to vaccinate their children could not complete their vaccinations due to shortage of the vaccine in the market. This is due to the logistical process in pharmacies, where inventory dependencies were not established between first and second portions of the vaccines. The shortage of vaccines invalidated the whole vaccination process for these patients. That is, the actual process flow was different from the model of the local process flow: step 8b (in Fig. 1) was not always reached.

GPM-based analysis: In GPM terms, state S5 was not reached for the second time, and the process was “stuck” in state S12. Surprisingly, S12 was identified as an unstable state. This implied that one of the following was occurring:

Option (1): S12 is a stable state and an external event is not being delivered, or,
Option (2): the model is missing a state between states S12 and S5 (steps 12b and 8b).

Considering the vaccine purchase within the process domain, option (2) is the valid one for the current case. The vaccine shortage is a newly discovered stable state. Note that we could consider the market as being in the environment of the process domain. Then S12 would be a stable state, awaiting an external event (vaccine purchasing).

Suggesting and evaluating possible modifications to the process model: First, the learning process can modify the modeled local system to reflect the actual flow,

turning step 12 and 12b into decision points and adding a new flow into a new step 17 "market shortage" in addition to the normal flow into step 8b. In GPM terms, this results in a new undesired stable state. However, this modification by itself did not solve the problem, as parents did not wish to vaccinate their other children after their bad experiences. Nevertheless, the modification made the problem explicit and understood, so two manual solutions could be proposed:

(1) Modifying the external environment processes in order to eliminate this exception. In our case, the inventory management processes of the clinic's pharmacy were not controlled by the clinical teams and therefore this solution was not feasible. External pharmacies were willing to adapt their processes only if the clinic could give them exclusivity agreements, which contradicted current legislations.

(2) Modifying the internal process flow to eliminate the stable state. To do so, the clinic required parents to purchase both immunization portions before administering the first portion. The second portion would be stored at the clinic until its administration, ensuring that the immunization process will be completed and thus eliminating the undesirable situation of portion 2 shortage. The modified process, including the automatically learned state and manual modification, is presented in Fig. 2. Note that shortage is still possible, but either a patient gets both portions, or none.

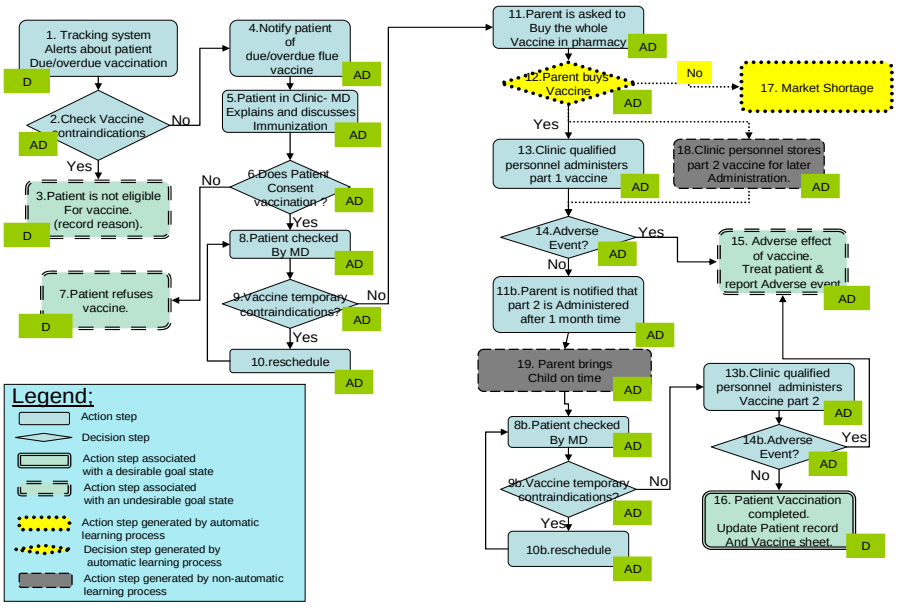

Fig. 2. Flu vaccine process modified following the learning and external changes. Note the automatic learning process has generated a new process step (step 17-market shortage) and detected a transition to it from step 12 (converted to a decision step). Note also the effect of non-automatic learning - elimination of step 12b (Parent buys part 2 vaccine) and the addition of steps 18 and 19.

4. Discussion

We illustrated how BP learning can be established based upon evaluation of soft-goals and exceptional states in process instances, and proposal of changes (manual, automatic, or semi-automatic) to the process model in order to minimize exceptions and to improve soft-goal accomplishment levels.

Examining the literature, we found several process model adaptation approaches proposed in the general BPM field [2, 4, 17, 18] as well as in medical guidelines modeling [2, 6]. Different process mining methods have been already discussed and applied in the BPM field [1, 4, 17] and in the clinical guidelines research [5, 19]. Comparison of a defined process model to the actual one discovered has been proposed [7, 19], as well as a statistic-based approach to propose ad-hoc changes both at process instance level an at process model levels [2, 3]. However, as far as we know, the examined literature is focused on understanding the actual control flow, without relating it to the process outcomes. Furthermore, it does not provide any methodology for defining the data needed for extending process mining to address both the process and its outcomes**.** Examining the process states definition of Table 2, we can easily identify the minimum EMR data needed to identify the process current state and history at any moment. Moreover, as state transitions cause changes in a subset of the domain state variables, tracking these state variables is sufficient for tracking the process state. Hence, the LPM can provide the data model required for mining a more holistic view of the actual process through the identification of the required state variables and the according documenting process. This is independent of which process mining algorithm is used and whatever process runtime model is used. We consider this to be a considerable advantage of LPM.

Learning is an evolutionary process, whose initial model is presented in this paper. As we execute the process we may face several situations which the learning process must address:

(1) Incomplete process model specification, i.e., unidentified states, incomplete state definitions, missing external events, and potential exceptions that invalidate the process execution.
(2) Causality relationships of selected process paths and achieved soft-goal values.
(3) Progress of new knowledge in the field, novel technology, or process innovations, which require changing the process

We must aim to provide the required changes at a process model proactively. To this end, LPM proposes a learning lifecycle-model based upon the process-goal approach. The contributions that we have made so far include: (a) A formal business process model-based approach that identifies the process execution state and how it attains its goals at any time. (b) A methodology for identifying what data sets need to be collected as part of process mining in order to enable process-learning. (c) A learning approach that introduces changes to the process model based upon identified exceptions, missing external events, and attainment of goals and soft-goals (the data sets described in (b)). The learning demonstrated in this paper relates to exception occurrences, while learning in general should also relate to soft-goal attained as a result of path selection. This will require the application of learning algorithms, which we intend to develop as future research. Note that while some learning may be done automatically, the modification of the business process may require, in no-error

tolerant applications (such as health care), some sort of expert confirmation before applying the modification. This does not contradict our approach.
Further work must be done to analyze potential automatic updates to the business process model. Another issue yet to be investigated is how knowledge integration could be accomplished based upon LPM.

References

1. Aalst WMP vd, Guenther C, Recker J, Reichert M. Using Process Mining to Analyze and Improve Process Flexibility - Position Paper. In: Proceedings 18th International Conference on Advanced Information Systems Engineering Workshops - BPMDS'06; 2006; 2006. p. 168-177.
2. Reichert M, Dadam P. ADEPT-flex: supporting dynamic changes of workflows without losing control. JIIS 1998;10(2):93-129.
3. Rinderle S, Reichert M, Dadam P. Correctness criteria for dynamic changes in workflow systems - a survey. Data & knowledge Engineering 2004;50:9-34.
4. Weske M. Formal foundation and conceptual design of dynamic adaptations in a workflow management system. In: Proceedings of the 34th Annual Hawaii International Conference on System Sciences; 2001; 2001. p. 7051.
5. Quaglini S, Stefanelli M, Lanzola G, Caporusso V, Panzarasa S. Flexible Guideline-based Patient Careflow Systems. Artificial Intelligence in Medicine 2001;22:65-80.
6. Peleg M, Kantor R. Approaches for guideline versioning using GLIF. In: Proc AMIA Symp.; 2003; 2003. p. 509-13.
7. Peleg M, Tu SW, Bury J, Ciccarese P, Fox J, Greenes RA, et al. Comparing Computer-Interpretable Guideline Models: A Case-Study Approach. J Am Med Inform Assoc 2003;10(1):52-68.
8. Advani A, Lo K, Shahar Y. Intention-Based Critiquing of Guideline-Oriented Medical Care. In: Proc. AMIA Annual Symposium; 1998; 1998. p. 483-487.
9. Soffer P, Wand Y. On the Notion of Soft Goals in Business Process Modeling. Business Process Management Journal 2005;11(6):663-679.
10. Soffer P, Wand Y. Goal-driven Analysis of Process Model Validity. In: Advanced Information Systems Engineering (CAiSE'04) (LNCS 3084); 2004; 2004. p. 521-535.
11. Bunge M. Treatise on Basic Philosophy: Vol. 4, Ontology II: A World of Systems. Boston: Reidel; 1979.
12. Bunge M. Treatise on Basic Philosophy: Volume 3: Ontology 1: The furniture of the world. Boston: Reidel; 1977.
13. Wand Y, Weber R. An Ontological Model of an Information System. IEEE Transactions on Software Engineering 1990;16(11):1282-1292.
14. Wand Y, Weber R. On the Ontological Expressiveness of Information Systems Analysis and Design Grammars. Journal of Information Systems 1993;3:217-237.
15. Oppenheim AV, Willsky AS. Signals & Systems, Second edition: Prentice Hall; 1997.
16. Institute for Clinical systems Improvements. Immunization, Eleventh edition,. In; 2006.http://www.icsi.org/Immunizations
17. Weber B, Reichert M, Rinderle S, Wild W. Towards a framework for the agile mining of business processes. In: Proceedings of BPM 05 BPI workshop; 2005; 2005.
18. Aalst WMPvd, Dongen BF, Herbst JL, Maruster L, Schimm G, Weijters AJMM. Workflow Mining: A Survey of Issues and Approaches. Data and Knowledge Engineering 2003;47(2):237-67.
19. Guth V, Oberweis A. Delta analysis of Petri-net based models for business processes. In: Proceedings of Applied Informatics; 1997. p. 23-32.

Mining Process Execution and Outcomes – Position Paper

Mor Peleg[1], Pnina Soffer[1], Johny Ghattas[1]

[1]Department of Management Information Systems, University of Haifa, Israel, 31905
morpeleg, pnina @{mis.hevra.haifa.ac.il}

Abstract. Organizational processes in general and patient-care processes in particular, change over time. This may be in response to situations unpredicted by a predefined business process model (or clinical guideline), or as a result of new knowledge which has not yet been incorporated into the model. Process mining techniques enable capturing process changes, evaluating the gaps between the predefined model and the practiced process, and modifying the model accordingly. This position paper motivates the extension of process mining in order to capture not only deviations from the process model, but also the outcomes associated with them (e.g., patient improving or deteriorating). These should be taken into account when modifications to the process are made.

Keywords: Process mining, business process modeling, Clinical guidelines

1. Introduction

Organizations often try to improve the quality of services provided to their clients by specifying a business process model (BPM) that captures the desired workflow in the organization and how exceptional situations should be handled. The aim is that if these process models would be implemented, the quality of services would increase and costs would be saved. In the healthcare domain, special attention has been given to creating evidence-based clinical guidelines that recommend care processes for patients with defined clinical conditions. Research has been carried out in developing methodologies and tools for specifying guidelines as computer-interpretable algorithms [1], linking them to clinical databases, executing them, and evaluating the impact of these systems [2].

However, organizational processes in general, and patient-care processes in particular, change over time. In real life, we often encounter situations that we neglected to consider in our BPM (e.g., guideline model). These situations could require process paths that were not specified in the BPM. In addition, expanding medical (or in general, organizational) knowledge on new procedures, available treatment options, and evidence for their effectiveness, may push users and

organizations into changing their process implementations, often before updating the BPM. Thus, we may find that the actors participating in the business processes (patient-care processes) often do not follow the BPM exactly and may act differently than the model specifies, which may or may not be justified or helpful.

Deviations of the actual performed process from its model have been studied both in the BPM community and in the Medical Informatics community. In the BPM community, process mining has been used to capture process changes, evaluate the gaps between the predefined BPM and the practiced process, and modifying the BPM accordingly [3, 4].

In the Medical Informatics community, critiquing approaches [5-7] have been used to compare the actual processes executed to their specified model. Advani et al. [7] describe a model and algorithm for deriving structured quality indicators and auditing protocols from formalized specifications of guidelines used in decision support systems. This critiquing approach can be used to determine whether the deviations followed the intentions of the original model and thus were justified. Traum-AID [6] is a rule-based system combined with a planner. Its critiquing interface examines actions the physician intends to carry out, identifies errors and calculates their significance, and produces a critique in response to those intentions. In the 1980's, Miller [5]developed several critiquing systems in which the physician inputs medical information describing a patient, a current set of test results, and current actions (e.g., ventilator settings), and a proposed set of new actions. The system assesses appropriate treatment goals, and uses those goals for critiquing.

Similar analysis can be used to offer decision-support to physicians only when they deviated from these intentions. Quaglini et al. [8] developed computerized guideline implementations that allow users not to follow al the actions specified in the model, justify the deviation, and select alternative activities out of a wide range of activities that were not planned by the guideline authors but that are related to the original alternative via hierarchies taken from standard clinical vocabularies. Similarly, many computerized guideline formalisms have tools that allow the user to deviate from the normal sequence of activities [9], as flexibility is often needed when the modeled guideline is to handle emergency situations in patients or when the model is out of date and does not convey the latest medical knowledge. Peleg and Kantor [10] used process mining to automatically analyze differences between two versions of process models that were created due to the expansion of medical knowledge, in order to find differences in particular medical knowledge concepts (e.g., new drug) or in concept relationships (e.g., pathogen is not longer believed to cause a disease).

As we strive to improve our BPMs (patient-care models), we must not only track deviations from the process model, but also the outcomes associated with them (e.g., patient improving or deteriorating) so that these could be taken into account when modifications to the process are made.

2. Background

In order to follow process outcomes and relate them to changes in the process, we need a formal process model that can represent goals and outcomes. We rely on the Generic Process Model (GPM) proposed by Soffer and Wand [11].

2.1 The Generic Process Model (GPM)

GPM is a state-based view of a process including the concept of goals. Briefly, GPM offers a process model which is composed of a quadruple <S, L, I, G>, where S is a set of states representing the domain of the process. Each state in an enacted process holds the values of all the properties (or state variables) of the process domain at a moment in time. The law L specifies possible state transitions as mapping between subsets of states, defined by conditions over values of the domain state variables; I is a subset of unstable states, which are the initial states of the process after a triggering external event has occurred; G is a subset of stable states on which the process terminates, termed the goal of the process.

The process goal as addressed by GPM is a state meeting the conditions that should be achieved by the process. GPM distinguishes process goals from soft-goals, which are defined as an order relation on goal states [12]. In other words, soft-goals relate to the *desirability* of possible states in the goal set (all meeting the condition that terminates the process) according to defined business objectives. For example, the goal of a process may be a state where some treatment has been given to a patient, but a state where the treatment does not incur side effects is considered as "better" than a state where side effects are observed. Finally, GPM entails criteria for assessing the validity of a process, namely, its ability to achieve its goal [11]. It enables the analysis of a process to identify causes for invalidity, and suggests appropriate redesign actions to eliminate these causes.

For operational and representational purposes, GPM's law can be mapped to Petri Nets [13]; states correspond to sets of places of the Petri Net and laws correspond to transitions (including transition guards). Complementing this representation, GPM's clear distinction of goals and soft-goals can form a basis for improving a practiced process, where improvement can be related to attained soft-goal values and to fewer situations where the goals of the process are not met.

3. Research Objectives

Objective 1: Develop a method for establishing process data on which outcome and goal analysis will be based. This includes patient-specific data referred to by the process model (e.g., age), data about activities that were started and completed, and data regarding outcomes, judged by relevant soft-goal attainment.

Objective 2: Develop methods and tools for analyzing process data to identify relationships between patient-specific data, execution paths, process goals, and outcomes.

4. Demonstration of Our Approach

To demonstrate and motivate our approach, we use a process model based on a guideline for treatment of ear infections (acute otitis media, AOM) [14]. Figure 1 shows a Petri Net of the process model adapted from that guideline. Such a Petri Net can be automatically converted from a GLIF algorithm [15].

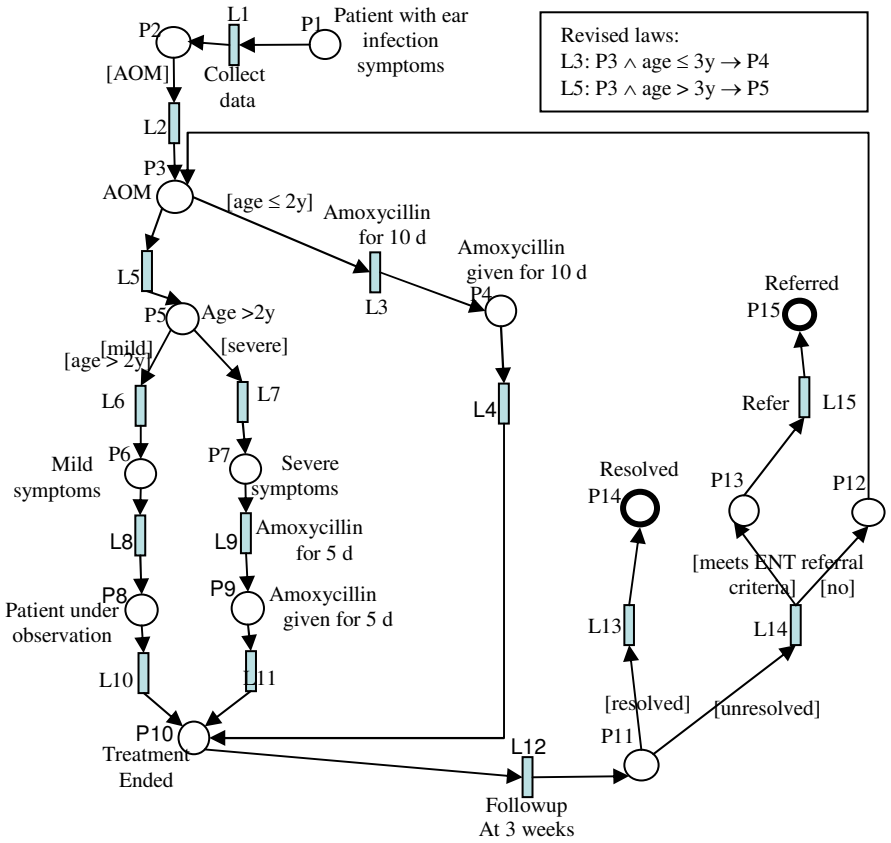

Figure 1. A Petri Net representation of an ear infection clinical algorithm. Places, corresponding to GPM states, are marked as Pi. Transitions, corresponding to GPM laws, are marked as Li. P14 is the desired goal state, P15 is the undesired goal state, and P1 is the initial state.

In order to have a measure of process attainment of soft and hard goals, and also of exceptions, where goals are not met, we need to analyze and mine the execution

log of the actual process, or their reflection in electronic medical records (EMRs), to document for each work item (i.e., an activity performed by an actor on a given case). In addition to the existing process mining ability to identify that an activity has been performed by an actor at a certain time, we also need to mine changes to state variables (e.g., a patient's temperature or his adverse response to a drug) that were not predicted by the process model, and their timestamp. These data could help us in relating activities (both those that followed the process model and those that represent changes) to outcomes. Table 1 presents a potential EMR of a 2.4 year-old patient reflecting the ear-infection process instance as well as outcomes. As can be seen, the physician first followed the guideline, prescribing Amoxycillin for 5 days, as the patient was over 2 years old. But, when AOM was not resolved and the goal state was not reached, he decided to prescribe a 10-day Amoxycillin, which was not according to the guideline. This time, the goal state was reached. Analysis of EMR data of other patients showed that in many of the 2-3 year old patients, AOM was not resolved after 5-day treatment. These relationships between goals and outcomes could suggest ways to improve the clinical process. For example, change the laws L3 and L5 such that patient under 3 (not under 2) would receive a 10-day treatment (Figure 1).

Table 1. EMR reflecting an ear-infection treatment process and outcomes

Time-Date	State variable	Value	Petri Net place
07-01-01:08:00	AOM	mild	P6
07-01-01:08:10	Medication	5-day Amoxycillin	P9
07-01-21:08:00	AOM	mild	P6
07-01-21:08:10	Medication	10-day Amoxycillin	P4
07-02-12:08:00	AOM	false	P14

5. Future Work

While the above ear infection scenario demonstrates the potential contribution of process execution and outcome analysis, a systematic method for such analysis is still under development. We are currently working on a hypothetical case study, taken from the clinical guideline domain of vaccinations, which examines actual execution or processes (as determined from synthetic EMR records). We will study how we can automatically deduce from the EMR records whether instances of the process have attained the process model's soft and hard goals, and when and how to characterize exceptional situations. We would like to use data about real process execution and outcomes (log files) along with the preconceived process models to test whether our approach could be used to automatically assess attainment of soft and hard goals as well as assess the occurrence of an exception (i.e., the invalidation of the process due to an unexpected event), resulting in processes that do not meet their goals and remain in intermediate states. We would then like to combine outcomes analysis with delta analysis for populations of patient with similar characteristics to suggest a linkage between process changes and process outcomes.

The main challenge we are facing is how to establish a causal relationship between the execution data (or delta analysis) and the obtained outcomes. The outcomes of

clinical processes are determined not only by the actions taken, but also by pre-existing patient properties, such as age in the ear infection example. The analysis should take these properties into account as affecting variables, and provide recommendations with respect to specific patient properties.

References

1. Peleg M, Tu SW, Bury J, Ciccarese P, Fox J, Greenes RA, et al. Comparing Computer-Interpretable Guideline Models: A Case-Study Approach. J Am Med Inform Assoc 2003;10(1):52-68.
2. Garg AX, Adhikari NKJ, McDonald H, Rosas-Arellano MP, Devereaux PJ, Beyene J, et al. Effects of Computerized Clinical Decision Support Systems on Practitioner Performance and Patient Outcomes: A Systematic Review. JAMA 2005;293(10):1223-1238.
3. Aalst WMPvd, Dongen BF, Herbst JL, Maruster L, Schimm G, Weijters AJMM. Workflow Mining: A Survey of Issues and Approaches. Data and Knowledge Engineering 2003;47(2):237-67.
4. Reichert M, Dadam P. ADEPT-flex: supporting dynamic changes of workflows without loosing control. JIIS 1998;10(2):93-129.
5. Miller PL. Goal-directed critiquing by computer: ventilator management. Comput Biomed Res 1985;18(5):422-38.
6. Gertner AS. Plan Recognition and Evaluation for On-line Critiquing. User Modeling and User-Adapted Interaction 1997;7(2):107-140.
7. Advani A, Goldstein M, Shahar Y, Musen MA. Developing Quality Indicators and Auditing Protocols from Formal Guideline Models:. In: AMIA Annual Symp.; 2003; 2003. p. 11-15.
8. Quaglini S, Stefanelli M, Lanzola G, Caporusso V, Panzarasa S. Flexible Guideline-based Patient Careflow Systems. Artificial Intelligence in Medicine 2001;22:65-80.
9. Peleg M. Guideline and Workflow Models. In: Greenes RA, editor. Clinical Decision Support - The Road Ahead: Elsevier/Academic Press; 2006.
10. Peleg M, Kantor R. Approaches for guideline versioning using GLIF. Proc AMIA Symp. 2003:509-13.
11. Soffer P, Wand Y. Goal-driven Analysis of Process Model Validity. In: Advanced Information Systems Engineering (CAiSE'04) (LNCS 3084); 2004; 2004. p. 521-535.
12. Soffer P, Wand Y. On the Notion of Soft Goals in Business Process Modeling. Business Process Management Journal 2005;11(6):663-679.
13. Peterson JL. Petri Net Theory and the Modeling of Systems. Englewood Cliffs, NJ: Prentice-Hall; 1981.
14. Institute for Clinical Systems Improvement. Diagnosis and Treatment of Otitis Media in Children. In; 2004.
15. Peleg M, Tu S, Mahindroo A, Altman R. Modeling and Analyzing Biomedical Processes using Workflow/Petri Net Models and Tools. MedInfo 2004:74-8.